Special Edition

MS-DOS® USER'S GUIDE

Most Complete DOS Coverage!

Covers Versions 3.X & 4.0

EDLIN text editor

DOS Shell

MS-DOS & PC DOS

que

MS-DOS® User's Guide

Special Edition

Developed by Que Corporation

QUE® CORPORATION
LEADING COMPUTER KNOWLEDGE

MS-DOS® User's Guide

Special Edition

Copyright © 1989 by Que® Corporation

All rights reserved. Printed in the United States of America. No part of this book may be used or reproduced in any form or by any means, or stored in a database or retrieval system, without prior written permission of the publisher except in the case of brief quotations embodied in critical articles and reviews. Making copies of any part of this book for any purpose other than your own personal use is a violation of United States copyright laws. For information, address Que Corporation, 11711 N. College Ave., Carmel, IN 46032.

Library of Congress Catalog No.: 89-62434

ISBN: 0-88022-505-X

This book is sold *as is*, without warranty of any kind, either express or implied, respecting the contents of this book, including but not limited to implied warranties for the book's quality, performance, merchantability, or fitness for any particular purpose. Neither Que Corporation nor its dealers or distributors shall be liable to the purchaser or any other person or entity with respect to any liability, loss, or damage caused or alleged to be caused directly or indirectly by this book.

92 91 90 89 8 7 6 5 4 3 2 1

Interpretation of the printing code: the rightmost double-digit number is the year of the book's printing; the rightmost single-digit number, the number of the book's printing. For example, a printing code of 89-1 shows that the first printing of the book occurred in 1989.

MS-DOS User's Guide, Special Edition, may be used with DOS V3 and V4.

ABOUT THE AUTHORS

David W. Solomon is a Que product development specialist and staff writer who specializes in operating systems, database management, and programming. Since 1978, he has worked with a variety of operating systems on both minicomputers and microcomputers, and he has developed and written custom software for scientific and business applications on a world-wide basis. He was instrumental in determining the emphasis for this book, assembled the team of contributors, and worked closely with them through manuscript development, editing, and final review. In addition to being the primary author of this book, he is also the author of *MS-DOS QuickStart*, the coauthor of *dBASE IV Applications Library*, 2nd Edition, and a contributing author of *Using Turbo Prolog*, all published by Que Corporation.

Caroline M. Halliday is an electrical engineer with High Tech Aid in the Chicago area. Her company specializes in technical documentation and teaching for the PC environment. She has a Bachelor of Science (Hons) degree from the University of Manchester, England, and is a former technical editor for *PC Tech Journal*. She is coauthor of *IBM PS/2 Technical Guide*, published by Howard W. Sams and Co., and the principal author of *Using OS/2*, published by Que Corporation. She now contributes articles to various magazines including *PC Week* and *Byte*.

David D. Busch is the author of 24 books, including four best-sellers that concentrate on MS-DOS. Most recently, he revised Que's highly respected *Using PC DOS*, 3rd Edition. The first two-time winner of "Best Book" honors from the Computer Press Association, he has served as a contributing editor and monthly columnist for six different computer magazines.

Timothy S. Stanley has worked for Que since 1985 as a technical editor. In addition to serving as technical editor for a number of books published by Que, he has contributed to *Using 1-2-3 Release 2.2*, Special Edition; *Using 1-2-3 Release 3*; *Upgrading to 1-2-3 Release 3*; and *Absolute Reference: The Journal for 1-2-3 and Symphony Users*.

Publishing Director

David Paul Ewing

Product Development Director

David W. Solomon

Acquisitions Editor

Terrie Lynn Solomon

Production Editor

Mary Bednarek

Editors

Kathie-Jo Arnoff
Tim Huddleston
Lisa Hunt
Jeannine McDonel
Virginia Noble
Alice Martina Smith

Technical Editors

Caroline M. Halliday
Jeff Sabotin
Timothy S. Stanley
Joyce J. Nielsen

Editorial Assistant

Stacey Beheler

Indexers

Sharon Hilgenberg
Joelynn Gifford

Book Design and Production

Dan Armstrong
Brad Chinn
Lori Lyons
Jennifer Matthews
Diana Moore
Mitzi Parsons
Jon Ogle
Joe Ramon
Patricia Maupin Riggs
Dennis Sheehan
Joy Shepherd
Mary Beth Wakefield
Graphics by William Hartman

Composed in Garamond and Universal Monospace No. 2 by Que Corporation

CONTENTS AT A GLANCE

Part I Understanding the Fundamentals of DOS

Chapter 1 Understanding the Role of DOS 11
Chapter 2 Understanding the DOS File System 27
Chapter 3 Understanding the Command Processor 69
Chapter 4 Gaining Control of the Hardware 97

Part II Working with the Core Commands

Chapter 5 Preparing and Maintaining Disks 121
Chapter 6 Managing and Navigating the Directory Hierarchy 181
Chapter 7 Managing Your Files 231
Chapter 8 Working with System Information 303
Chapter 9 Gaining Control of Devices 333

Part III Expanding Your Use of DOS

Chapter 10 Using EDLIN 373
Chapter 11 Understanding Batch File Basics 399
Chapter 12 Understanding ANSI.SYS 417
Chapter 13 Learning Advanced Batch File Techniques 431

Part IV Advancing Your DOS Capabilities

Chapter 14 Configuring Your Computer 455
Chapter 15 Understanding the International Features of DOS 497
Chapter 16 Using the DOS Shell 527

Part V Command Reference (Includes Messages Section)

Introduction to the Command Reference 579

Appendix A Changes between Versions of DOS 791
Appendix B ASCII and Extended ASCII Codes 801
Appendix C DOS Control and Editing Keys 813
Appendix D Preparing Your Hard Disk 817
Appendix E ANSI Terminal Codes 829
Appendix F Code Page Tables 837

Index 843

TABLE OF CONTENTS

Introduction 1

Who Should Read This Book? 1
What Hardware Is Needed? 2
What Versions of DOS Are Covered? 2
What Is Not Covered? 2
The Details of This Book 3
Learning More About DOS 6
Conventions Used in This Book 7

I Understanding the Fundamentals of DOS

1 Understanding the Role of DOS 11

What is DOS? 12
The Variations of MS-DOS 12
Understanding DOS in Context 14
The DOS Package 14
The Bootable Operating System Disk 16
The System Prompt 17
The Hidden Core of DOS 18
The Configuration Capability of DOS 19
Examining the Layers of DOS 19
The Hardware 21
The Basic Input/Output System (BIOS) 22
The DOS Kernel 23
The Command Processor (COMMAND.COM) 24
The DOS Shell 24
Applications Programs 24
Summary 25

2 Understanding the DOS File System 27

Examining Files and File Systems 28
Understanding Files 28
Understanding the File System 29
Examining Disks and Drives 30
Understanding the Disk's Magnetic Storage Technique 31
Understanding Disk Drives 32
Hard Disk Drives 32
Floppy Disk Drives 33

Understanding the Dynamics of the Disk Drive 35
Disk Drive Heads 35
Disk Tracks 35
Cylinders 36
Seek Time 37
Disk Drive Alignment 38
Applying Drive Dynamics to Data Storage 39
Track Density 39
Data Density 40
Disk Sectors 41
Disks as Binary Storage 42
Bytes and Disk Capacity 43
Understanding Disk Formats 44
Floppy Drive Formats 44
Raw Capacity and Usable Capacity 46
Hard Disk Drive Formats 46
Examining How DOS Manages Disks and Files 47
Understanding Storage Allocation Logging 48
Understanding Coordinates and Logical Sectors 52
Knowing What DOS Stores—the Directory 53
Understanding the Root Directory and its Subdirectories 58
Subdirectories as Files 59
A Hierarchy of Directories 59
Directory Paths 61
Understanding Defaults 61
Understanding Disk Partitions 62
Looking at the DOS File System from the Ground Up 65
Summary 67

3 Understanding the Command Processor 69

Examining the Role of Hardware 69
Peripheral Hardware 70
System-Unit Hardware 70
The Microprocessor 71
Memory 73
Buses and Control Chips 74
Programming Languages 74
Booting DOS 75
Understanding Commands 77
Views of Command Names 77
Categories of DOS Commands 78
Understanding the Command Line 85

Understanding Parameters 85
Understanding Switches 87
Understanding Defaults 88
Using the Keyboard 89
Stopping a Command 93
Understanding the Environment 93
Summary 94

4 Gaining Control of Hardware 97

Reviewing the Layered PC 97
Understanding Device Drivers 98
Understanding Controllers 99
Grouping Device Drivers 100
Character Device Drivers 100
Block Device Drivers 101
Defining Resident and Installable Device Drivers 102
Resident Device Drivers 102
Installable Device Drivers 104
Understanding CONFIG.SYS 106
Understanding Extended and Expanded Memory 108
Understanding Extended Memory 109
Understanding Expanded Memory 111
Understanding Standard Input and Output 113
Understanding Redirection 114
Understanding Piping 115
Summary 116

II Working with the Core Commands

5 Preparing and Maintaining Disks 121

Issuing External Commands 122
Dividing a Hard Disk with FDISK 122
Issuing the FDISK Command 123
Understanding Hard Disk Partitions 123
Understanding the Operation of FDISK 124
Formatting Disks with FORMAT 126
Issuing the FORMAT Command 126
Understanding the Operation of FORMAT 128
Formatting the Control Area 128
Managing Disks Larger than 32M 129
Creating the Root Directory 130
Formatting the Files Area 131

Writing a Volume Label 131
Copying the System Files 132
Understanding the General Rules of FORMAT 132
Formatting a Diskette 133
Formatting a Bootable Diskette with the DOS Shell 137
Formatting a Hard Disk 138
Completing Preliminary Steps 138
Issuing FORMAT for the Hard Disk 140
Naming Disks with LABEL 141
Issuing the LABEL Command 142
Understanding the Operation of LABEL 142
Understanding the General Rules of LABEL 143
Labeling a Disk 144
Examining Volume Labels with VOL 145
Issuing the VOL Command 145
Understanding the General Rules of VOL 145
Transferring the DOS System Using SYS 145
Issuing the SYS Command 146
Understanding the Operation of SYS 146
Understanding the General Rules of SYS 147
Using SYS To Transfer DOS to a Diskette 148
Copying Disks with DISKCOPY 148
Issuing the DISKCOPY Command 149
Understanding the Operation of DISKCOPY 149
Understanding the General Rules of DISKCOPY 151
Using DISKCOPY To Copy the DOS Startup Disk 151
Using DISKCOPY with Two Floppy Disk Drives 153
Using DISKCOPY with One Floppy Disk Drive 154
Labeling and Write-Protecting Your New Disks 156
Understanding DISKCOPY's Error Messages 156
Comparing Diskettes with DISKCOMP 157
Issuing the DISKCOMP Command 158
Understanding the Operation of DISKCOMP 158
Understanding the General Rules of DISKCOMP 158
Using DISKCOMP To Compare Diskettes 159
Selectively Updating a Disk with REPLACE 160
Issuing the REPLACE Command 160
Understanding the Operation of REPLACE 161
Understanding the General Rules of REPLACE 162
Using REPLACE To Update to a New Version of DOS . . 163
Upgrading to DOS V3.3 163
Upgrading to DOS V4 165
Analyzing a Disk with CHKDSK 165

Issuing the CHKDSK Command 165
Understanding the Operation of CHKDSK 166
Understanding the General Rules of CHKDSK 169
Using CHKDSK To Repair an Allocation Size Error 170
Recovering Files and Disks with RECOVER 172
Issuing the RECOVER Command 173
Using RECOVER on a Defective File 173
Issuing RECOVER To Recover a Single File 174
Using RECOVER on a Damaged Directory 176
Taking Preliminary Steps 176
Issuing RECOVER To Recover a Directory 176
Identifying the Recovered Files 177
Understanding the General Rules of RECOVER 178
Summary 178

6 Managing and Navigating the Directory Hierarchy 181

Understanding the Hierarchical Directory System 182
Understanding the Directory Entry 182
Understanding the Root Directory 184
Understanding Subdirectories 184
Specifying a Path in a Command Line 185
Using PROMPT To Display a Full Path 187
Making Directories with MKDIR (MD) 187
Issuing the MKDIR Command 188
Understanding the Operation of MKDIR 188
Understanding the General Rules of MKDIR 189
Creating a Directory with MKDIR 190
Changing the Current Directory with CHDIR (CD) 191
Issuing the CHDIR Command 191
Understanding the Operation of CHDIR 191
Understanding the General Rules of CHDIR 192
Using the Two Forms of CHDIR To Change Directories 192
Defining the Program Search with PATH 194
Issuing the PATH Command 194
Understanding the Operation of PATH 195
Understanding the General Rules of PATH 196
Using PATH To Search Directories 197
Using Aliases with PATH 198
Simplifying Data-File Searching with APPEND 199
Issuing the APPEND Command 200
Issuing the V3.2 APPEND Command 201

Issuing the V3.3 APPEND Command 202
Issuing the V4 APPEND Command 203
Understanding the Operation of APPEND 204
Understanding the General Rules of APPEND 205
Using APPEND with the /X Switch 206
Listing Directories with TREE 208
Issuing the TREE Command 209
Understanding the Operation of TREE 209
Using TREE To View a Disk's Hierarchical Structure 211
Comparing CHKDSK and TREE 213
Controlling the Output of TREE with Filters and Redirection 214
Combining Copying and Directory Management with XCOPY 215
Issuing the XCOPY Command 215
Understanding XCOPY's Switches 216
Understanding the Operation of XCOPY 218
Understanding the General Rules of XCOPY 218
Using XCOPY Effectively 219
Removing an Unneeded Directory with RMDIR (RD) 220
Issuing the RMDIR Command 220
Understanding the Operation of RMDIR 220
Understanding the General Rules of RMDIR 221
Deleting an Empty Directory with RMDIR 221
Taking Advantage of the Hierarchical Directory System . . . 224
Keeping a Clean Root 224
Including a Directory To Store DOS Programs 225
Ensuring Uniform Directories for Applications Programs 226
Using a Temporary Directory 227
Keeping Files in \KEEP 228
Summary 229

7 Managing Your Files 231

Reviewing the Concept of Files 232
Selecting File Names 232
Avoiding Reserved Names 235
Observing File-Naming Conventions 236
Avoiding Bad File Names 238
Finding the End of a File 239
Understanding Devices and the End-of-File Marker 241
Manipulating File Attributes with ATTRIB 242
Issuing the ATTRIB Command 242

Understanding the Operation of ATTRIB 243
Establishing Read-Only Files 243
Displaying Archive Attributes 244
Understanding the General Rules of ATTRIB 244
Using Archive Attributes with XCOPY, BACKUP, and RESTORE 244
Using ATTRIB To Write-Protect Applications Program Files 246
Using ATTRIB with the XCOPY Command 247
Making COMMAND.COM Read-Only with ATTRIB 250
Changing File Names with RENAME (REN) 251
Issuing the RENAME Command 251
Understanding the Operation of RENAME 252
Understanding the General Rules of RENAME 252
Renaming Files 253
Deleting Files with ERASE (DEL) 256
Issuing the ERASE Command 257
Understanding the Operation of ERASE 257
Understanding the General Rules of ERASE 258
Using ERASE with the ATTRIB Command 258
Viewing a File's Contents with TYPE 260
Issuing the TYPE Command 260
Understanding the Operation of TYPE 260
Understanding the General Rules of TYPE 261
Viewing the Contents of CONFIG.SYS with TYPE 262
Using TYPE and Redirection 262
Comparing Files with COMP 263
Issuing the COMP Command 263
Understanding the Operation of COMP 264
Understanding the General Rules of COMP 265
Comparing Two Text Files with COMP 266
Performing a Full-File Comparison with FC 268
Issuing the FC Command 268
Understanding the Operation of FC 269
Understanding the General Rules of FC 270
Using FC To Compare Two Versions of a Memo 271
Tapping the Power of COPY 272
Issuing the COPY Command 273
Understanding the Operation of COPY 274
Understanding the General Rules of Copy 276
Respecting the COPY Command 277
Using COPY in a Variety of Operations 278
Copying from Directory to Directory 278
Copying from Disk to Disk 280

Concatenating Text Files 281
Copying from a Device to a File 283
Concatenating a File and a Device to a File 284
Joining Files in Binary Mode 285
Turning Verification On with VERIFY 286
Issuing the VERIFY Command 287
Understanding the General Rules of VERIFY 287
Verifying a Critical BACKUP Operation with VERIFY . . . 288
Making Backup Sets with BACKUP 289
Issuing the BACKUP Command 290
Understanding the General Rules of BACKUP 292
Determining a BACKUP Policy 292
Issuing the Full BACKUP Command 293
Following Up with Incremental Backups 294
Issuing the Multiset Incremental Backup Command 296
Restoring Backed-up Files with RESTORE 296
Issuing the RESTORE Command 297
Understanding the Operation of RESTORE 298
Understanding the General Rules of RESTORE 299
Using RESTORE Following a Disk Failure 299
Using RESTORE Following an ERASE *.* Command 300
Summary 301

8 Working with System Information 303

Changing the Date and Time with DATE and TIME 304
Issuing the DATE Command 306
Understanding the Operation of DATE 306
Issuing the TIME Command 307
Understanding the Operation of TIME 307
Displaying the DOS Version with VER 308
Issuing the VER Command 309
Understanding the Operation of VER 309
Checking Machine Configuration with MEM 309
Issuing the MEM Command 310
Understanding the Operation of MEM 310
Listing Files with DIR 313
Issuing the DIR Command 313
Understanding the Operation of DIR 313
Displaying a Screenful of Information with DIR 315
Using the DIR Command with Wild Cards 317
Using the Root Name of a File in a DIR Command 317
Using DIR Across Subdirectories 318

Accessing the DOS Shell User Interface with
DOSSHELL 318
Issuing the DOSSHELL Command 319
Understanding the Operation of DOSSHELL 319
Changing the Command Prompt with PROMPT 319
Issuing the PROMPT Command 320
Understanding the Operation of PROMPT 321
Changing the Character Set with CHCP 321
Issuing the CHCP Command 322
Understanding the Operation of CHCP 322
Altering the Look of the Screen with MODE 322
Issuing the MODE Command 323
Understanding the Operation of MODE 323
Using MODE on a Color Graphics Adapter 324
Using MODE To Adjust the Number of Columns
or Lines On-Screen 325
Adjusting the Number of Columns 325
Adjusting the Number of Lines 326
Starting the Command Processor with COMMAND 326
Issuing the COMMAND Command 327
Understanding the Operation of COMMAND 327
Using EXIT To Leave the Current Copy of the
Command Processor 328
Issuing the EXIT Command 328
Starting and Exiting Secondary Command Processors 328
Changing the Environment with SET 329
Issuing the SET Command 330
Changing the Environment Variables with SET 330
Setting Your Own Environment Variables with SET 331
Summary 331

9 Gaining Control of Devices 333

Clearing the Screen with CLS 334
Issuing the CLS Command 334
Understanding the Operation of CLS 334
Using CLS To Clean Up After Batch Files 334
Printing Graphics Screens with GRAPHICS 335
Issuing the GRAPHICS Command 335
Understanding the Operation of GRAPHICS 335
Understanding the General Rules of GRAPHICS 337
Using GRAPHICS To Print a Screen Image 338
Printing in the Background with PRINT 338
Issuing the PRINT Command 339

Understanding the Operation of PRINT 340
Understanding the General Rules of PRINT 341
Using PRINT To Print Several Files 342
Redirecting File I/O with ASSIGN 344
Issuing the ASSIGN Command 344
Understanding the Operation of ASSIGN 344
Understanding the General Rules of ASSIGN 344
Using ASSIGN To Reassign Drives 345
Joining Two Disk Drives with JOIN 345
Issuing the JOIN Command 345
Understanding the Operation of JOIN 346
Understanding the General Rules of JOIN 346
Using JOIN To Connect Drives 347
Substituting a Drive Name for a Path with SUBST 347
Issuing the SUBST Command 347
Understanding the Operation of SUBST 348
Understanding the General Rules of SUBST 348
Using SUBST To Reference a Path with a Drive Letter 349
Extending a Keyboard's Code Capability with KEYB 350
Issuing the KEYB Command 350
Understanding the Operation of KEYB 351
Understanding the General Rules of KEYB 351
Using KEYB To Enable Foreign Language Keys 352
Selecting a Different Console with CTTY 353
Issuing the CTTY command 353
Understanding the Operation of CTTY 353
Understanding the General Rules of CTTY 353
Making the Serial Port the Console with CTTY 354
Controlling Device Operations with MODE 354
Using MODE To Change Parallel Port Settings 354
Issuing MODE To Change Parallel Port Settings 354
Understanding the General Rules of MODE To Change Parallel Port Settings 355
Understanding the Operation of MODE with Parallel Ports 355
Using MODE To Print a Large File 356
Using MODE To Change the Serial Port 357
Issuing MODE To Adjust Serial Ports 357
Understanding the General Rules of MODE with Serial Ports 357
Understanding the Operation of MODE with Serial Ports 358

Using MODE To Set the Serial Port 358
Using MODE To Redirect a Parallel Port to a Serial Port 359
Issuing the MODE Command To Redirect Ports 359
Understanding the Operation of MODE To Redirect Ports 359
Understanding the General Rules of MODE To Redirect Ports 359
Using MODE To Redirect Ports 360
Using MODE with DOS V4 To Change the Typematic Rate 360
Issuing the MODE Command To Set the Typematic Rate 360
Understanding the Operation of MODE To Set the Typematic Rate 361
Using MODE To Set a Delay 362
Redirecting Input and Output to Other Devices with Redirection 362
Issuing the Redirection Commands 362
Understanding the General Rules of Redirection 363
Understanding the Operation of Redirection 363
Using Redirection To Print the Output of Commands 363
Introducing Filters (MORE, FIND, SORT) and Piping (|) 364
Buffering with the MORE Filter 365
Issuing the MORE Filter 365
Understanding the Operation of MORE 365
Understanding the General Rules of MORE 365
Using MORE To Pause the Screen 365
Finding Strings of Text with FIND 366
Issuing the FIND Filter 366
Understanding the General Rules of FIND 367
Understanding the Operation of FIND 367
Using FIND To Find Files On-Disk 367
Sorting Information with SORT 368
Issuing the SORT Filter 368
Understanding the General Rules of SORT 368
Understanding the Operation of SORT 369
Using SORT To Sort Subdirectories 369
Summary 370

III Expanding Your Use of DOS

10 Using EDLIN **373**

Starting EDLIN 374
Creating a Memo with EDLIN 374
Inserting Text 375
Listing Text 377
Editing a Line 378
Saving Text 379
Printing the Memo 379
Deleting Lines 380
Learning More about EDLIN's Features 381
Using EDLIN's /B Switch 381
Understanding How EDLIN Works with Files 381
Learning Special Editing Techniques 382
Using the Editing Keys 383
Using the EDLIN Commands 384
A—The Append Lines Command 386
W—The Write Lines Command 387
C—The Copy Lines Command 387
D—The Delete Lines Command 389
I—The Insert Lines Command 389
L—The List Lines Command 390
P—The Page Command 391
M—The Move Lines Command 392
Q—The Quit Edit Command 393
S—The Search Text Command 393
R—The Replace Text Command 394
T—The Transfer Lines Command 396
Summary 396

11 Understanding Batch File Basics **399**

Using Batch Files 400
Understanding The Basic Rules of Batch Files 401
Creating Simple Batch Files 401
Creating a Batch File To Format Diskettes 402
Creating a Batch File To Mark a Directory as "Home" 404
Understanding Batch File Parameters 405
Understanding More About Variable Markers 406
Constructing a Batch File Using Parameters 408
Using Additional Batch Subcommands 410
Introducing REM and More about ECHO 410

Controlling the Display with CLS 411
Branching with GOTO 411
Reviewing Batch File Rules 414
Summary 415

12 Understanding ANSI.SYS 417

Installing Device Drivers 417
Communicating with ANSI.SYS 419
Embedding Escape Characters in Files 419
Using PROMPT To Enter Escape Characters 420
Using the Cursor-Movement Commands 421
Using the Screen Commands 423
Using the Keyboard Commands 425
Understanding Scan Codes 425
Redefining Scan Codes 428
Summary 430

13 Learning Advanced Batch File Techniques 431

Reviewing Batch File Subcommands 431
Using IF 432
Using IF To Test ERRORLEVEL 433
Using IF To Compare Strings 433
Using IF To Look for Files 435
Using FOR..IN..DO 435
Using a FOR..IN..DO Batch File 436
Using FOR..IN..DO at the DOS Prompt 437
Using FOR..IN..DO with Wild Cards 437
Redirecting FOR..IN..DO 439
Using FOR..IN..DO with Other Commands 440
Moving Parameters with SHIFT 440
Running Batch Files from Other Batch Files 441
Calling a Batch File at the End of Another 442
Calling a Batch File with CALL 442
Understanding CALL Batch Files 442
Analyzing Results of CALL Batch Files 444
Calling a Batch File with COMMAND /C 445
Using the Special Batch File AUTOEXEC.BAT 446
Viewing a Sample AUTOEXEC.BAT File 447
Understanding the Sample AUTOEXEC.BAT File 447
Building Menus 449
Dressing Up the Menu with ANSI.SYS 450
Summary 451

IV Advancing Your DOS Capabilities

14 Configuring Your Computer 455

Understanding CONFIG.SYS 456
Telling DOS When To Break 457
Using BUFFERS To Increase Disk Performance 458
Understanding How DOS Uses Disk Buffers 459
Fine-Tuning with Buffers 460
Understanding DOS V4 BUFFERS Options 461
Installing Device Drivers with DEVICE 461
Understanding Disk Caches 462
Installing IBMCACHE.SYS 464
Changing the Cache Location 464
Changing the Cache Size 466
Changing the Cache Page Size 466
Changing the IBMCACHE Settings in CONFIG.SYS 468
Installing SMARTDRV.SYS 468
Using RAMDRIVE To Make a RAM Disk 469
Understanding RAMDRIVE Options 469
Learning More about RAM Disks 471
A RAM Disk in 640K of Memory 471
A RAM Disk in Expanded or Extended Memory 472
Controlling Additional Disk Drives with DRIVER.SYS 474
The DRIVER.SYS Switches 474
Sample DRIVER.SYS Directives 476
Naming Disk Drives 478
Using Expanded Memory with DOS V4 479
Using the XMA2EMS.SYS Driver 479
Using the XMAEM.SYS Driver 480
Displaying and Printing Different Languages 480
Displaying International Characters with DISPLAY.SYS 480
Printing International Characters with PRINTER.SYS 482
Accessing Files through FCBS 484
Using the FILES Directive 485
Using LASTDRIVE To Change the Number of Disk Drives 485
Using the SHELL Directive 486
Using the STACKS Directive 488
Understanding DOS V4 CONFIG.SYS Commands 489

Using the REM Directive 490
Using the SWITCHES Directive 490
Using the INSTALL Directive 490
Gaining More Speed with FASTOPEN 491
Making a CONFIG.SYS File 493
Summary 494

15 Understanding the International Features of DOS 497

Understanding Code Pages 498
Using the International Directives of DOS 500
Changing the CONFIG.SYS File 500
Adding the COUNTRY Directive 500
Adding DISPLAY.SYS 502
Adding PRINTER.SYS 504
Changing the AUTOEXEC.BAT File 507
Supporting National Languages with NLSFUNC 507
Changing the Code Page with CHCP 508
Selecting a Language for the Keyboard with KEYB . . . 509
KEYB for DOS V3.2 and Earlier Versions 509
KEYB for DOS V3.3 and V4 511
Changing the Code Page with MODE 512
Defining Code Pages with MODE CODEPAGE PREPARE 513
Activating Code Pages with MODE CODEPAGE SELECT 515
Refreshing a Device's Memory with MODE CODEPAGE REFRESH 517
Checking Code Pages with MODE /STATUS 517
Solving Display Problems with GRAFTABL 519
Using SELECT To Set Country Information 519
SELECT for DOS V3.2 and V3.3 520
A Sample COUNTRY Setup 522
Adding International Directives to CONFIG.SYS 522
Adding International Directives to AUTOEXEC.BAT 523
Summary 524

16 Using the DOS Shell 527

Getting To Know the Shell Menu Tree 527
Understanding the Shell Display Modes 529
Scrolling Text 531
Using the Help System 532
Using the Start Programs Menu 534

Using Function Keys 536
The F1 Key 536
The Shift-F9 Key 536
The F10 Key 536
Using the Program Submenu 537
Starting a Program 538
Adding a Program 538
Using Prompt Panels 540
Using Additional Add Program Parameters 542
Changing a Program 545
Deleting a Program 546
Copying a Program 546
Using the Group Submenu 547
Adding a Group 548
Changing a Group 548
Deleting a Group 549
Changing the Order of a Group's Listings 549
Creating a New Main Group Menu 549
Using the Exit Submenu 552
Using the Main Group Menu 552
Starting Programs 552
Using the Command Prompt 552
Using the File System Menu 552
Using the File System File Option 554
Opening a File 554
Printing a File 555
Associating a File 556
Moving a File 557
Copying a File 557
Deleting a File 559
Renaming a File 559
Changing a File's Attributes 559
Viewing a File 560
Creating a Directory 561
Selecting All Files 561
Deselecting All Files 561
Using the File System Options Option 561
Selecting Display Options 561
Selecting File Options 561
Showing File Information 562
Using the File System Arrange Option 563
The Multiple File List Option 563
The System File List Option 564

Using the File System Exit Option 564
Using Change Colors 564
Using the DOS Utilities Menu 565
Installing DOS V4 and the Shell with SELECT 565
Choosing a Configuration 566
Choosing Drive and Printer Information 567
Understanding a Sample DOSSHELL.BAT File 568
Customizing the Shell 569
The /MOS: Option 570
The /TRAN Option 570
The /COLOR Option 570
The /DOS Option 570
The /MENU Option 571
The /MUL Option 571
The /SND Option 571
The /MEU: Option 571
The /CLR: Option 571
The /PROMPT Option 572
The /MAINT Option 572
The /EXIT Option 572
The /SWAP Option 572
The /DATE Option 573
The /ASC: Option 573
The /B: Option 573
The /CO1, /CO2, and /CO3 Options 573
The /TEXT Option 574
The /COM2 Option 574
The /LF Option 574
Summary 574

V Command Reference

Introduction to the Command Reference 579

Notation 580
Notation for External Commands and Batch Files 580
Using Upper- and Lowercase Letters 581
DOS Messages 581
General DOS Messages 582
DOS Device Error Messages 595
APPEND 598
ASSIGN 603

ATTRIB 605
BACKUP 607
Batch Command 616
Batch Subcommand—CALL 618
Batch Subcommand—ECHO 619
Batch Subcommand—FOR..IN..DO 620
Batch Subcommand—GOTO 621
Batch Subcommand—IF 622
Batch Subcommand—PAUSE 623
Batch Subcommand—REM 624
Batch Subcommand—SHIFT 624
BREAK 625
CHCP 625
CHDR or CD 627
CHKDSK 629
CLS 638
COMMAND 639
COMP 640
Configuration Subcommand—BREAK 643
Configuration Subcommand—BUFFERS 644
Configuration Subcommand—COUNTRY 645
Configuration Subcommand—DEVICE 646
Configuration Subcommand—DRIVPARM 647
Configuration Subcommand—FCBS 648
Configuration Subcommand—FILES 649
Configuration Subcommand—INSTALL 649
Configuration Subcommand—LASTDRIVE 650
Configuration Subcommand—REM 650
Configuration Subcommand—SHELL 651
Configuration Subcommand—STACKS 652
COPY 652
CTTY 656
DATE 657
DEL 658
DIR 658
DISKCOMP 660
DISKCOPY 664
ERASE 670
EXE2BIN 672
EXIT 673
FASTOPEN 673
FC 676
FIND 679

FORMAT 681
GRAFTABL 688
GRAPHICS 690
JOIN 691
KEYB 695
LABEL 700
MEM 702
MKDIR or MD 705
MODE—LPTPORT 706
MODE—DISPLAY 708
MODE—COM PORT 710
MODE—REDIRECTION 711
MODE—CODE PAGE PREP 712
MODE—CODE PAGE SELECT 714
MODE—CODE PAGE REFRESH 716
MODE—CODE PAGE STATUS 717
MODE—KEY REPEAT 718
MODE—STATUS 719
MODE—GENERAL 719
MORE 726
NLSFUNC 727
PATH 729
PRINT 731
PROMPT 737
RECOVER 740
RENAME or REN 743
REPLACE 744
RESTORE 751
RMDIR or RD 757
SELECT 758
SET 763
SHARE 764
SORT 766
SUBST 769
SYS 773
TIME 775
TREE 776
TYPE 778
VER 779
VERIFY 779
VOL 781
XCOPY 781

Appendixes

A Changes between Versions of DOS 791

B ASCII and Extended ASCII Codes 801

C DOS Control and Editing Keys 813

D Preparing Your Hard Disk 817

E ANSI Terminal Codes 829

F Code Page Tables 837

Index 843

ACKNOWLEDGMENTS

MS-DOS User's Guide, Special Edition, is the result of the dedicated contributions of a team of true microcomputer publishing professionals. All of the people who worked many long, hard hours deserve special thanks.

Que Corporation owes special gratitude to contributing author Caroline M. Halliday for her instrumental assistance in developing the scope, direction, and content of *MS-DOS User's Guide*, Special Edition. Her work on this project as both a technical editor and author represents quality involvement of the highest standard. Caroline's acute knowledge of the end-user market has been of great assistance in making this a practical, hands-on instructional text and reference tool.

Que Corporation would also like to thank David Busch for his ability to quickly and efficiently generate text geared to the intermediate/advanced DOS market. His extensive knowledge of batch files, his insights into DOS, and his professional authoring skills make working with David a pleasure.

Tim Stanley, a Que Corporation "staffer," did a greatly appreciated, although sometimes thankless job of contributing key portions of the text. Tim's collation and refinement of the comprehensive Command Reference will benefit readers. When Tim was not busy writing or perfecting the Command Reference, he made himself available as a technical editor.

Publishing Director David P. Ewing provided the anchoring guidance, support, and talented direction that made this book possible.

Acquisitions Editor Terrie Lynn Solomon assembled a fine team of DOS experts for the most comprehensive DOS user's book available. During the project, Terrie acted as the command center of communication and coordination for the team of authors. The amount of material that passed through her office was impressive, but all of the material was perfectly coordinated.

Production Editor Mary Bednarek's work represents the finest example of Que editorial commitment. Mary organized the editing team, established the style and conventions for the book, and worked with the production department. Working with a professional like Mary is indeed a pleasure.

Editors Jeannine McDonel and Tim Huddleston fine-tuned the Command Reference through hundreds of pages of syntax conventions and still managed to remain cheerful. Their efforts are greatly appreciated.

Editor Alice Martina Smith showed her speed and efficiency by turning chapters in a timely and efficient manner, and editor Ginny Noble availed her skillful copy-editing talents to the project.

Kathie-Jo Arnoff used her editorial versatility to "pinch-hit" whenever needed, but especially "in the bottom of the ninth."

Lisa Hunt displayed dedication "above and beyond the call of duty" by doing a super editing job in the final, critical stages of the project.

Thanks also go to the Tech Editors for the project:

Caroline Halliday for tech editing chapters and providing key critical evaluation.

Jeff Sabotin for precise tech editing of the Command Reference.

Joyce Nielsen for tech editing the Lotus Magellan and XTreePro material.

Thanks to Stacey Beheler for editorial support throughout the project.

Thanks to Bill Hartman for the creation of graphics from our rough sketches.

Thanks must go to the Que production staff who took disk files and turned them into an accurate, attractive book:

Proofreaders David Kline, Lori Lyons, Diana Moore, Mitzi Parsons, and Joy Shepherd for their tireless, meticulous reading.

Typesetters Joe Ramon and Brad Chinn for typographic expertise.

Pasteup professionals Dennis Sheehan and Mary Beth Wakefield for their artful craftsmanship in pasting this book.

Designer Dan Armstrong for the creative and attractive layout.

Indexers Sharon Hilgenberg and Joelynn Gifford for precision and exactness.

And finally, Que owes deserved thanks to the vendors who supplied software and expertise to the team. Microsoft Corporation, and especially Tanya Van Dam, Public Relations, for her cooperation and support during the development and evolution of this project. Microsoft's assistance with MS-DOS and OS/2, has been a great help.

TRADEMARK ACKNOWLEDGMENTS

Que Corporation has made every attempt to supply trademark information about company names, products, and services mentioned in this book. Trademarks indicated below were derived from various sources. Que Corporation cannot attest to the accuracy of this information.

1-2-3, Blueprint, Lotus, Lotus Manuscript, and Magellan are registered trademarks of Lotus Development Corporation.

Apple, Macintosh, and Mac are registered trademarks of Apple Computer, Inc.

COMPAQ Personal Computer DOS Version 4.00 is a trademark, and COMPAQ and COMPAQ DeskPro 386 are registered trademarks of COMPAQ Computer Corporation.

CompuServe Information Service is a registered trademark of CompuServe Incorporated and H&R Block, Inc.

dBASE IV is a trademark, and dBASE III Plus is a registered trademark of Ashton-Tate Corporation.

DisplayWrite 4, IBM Personal Editor, IBM ProPrinter, IBM Token Ring, Micro Channel, OS/2, Systems Application Architecture (SAA), and TopView are trademarks, and IBM, PC XT, PS/2 Models 25, 30, 50, 50Z, 60, 70, and 80, and Quietwriter are registered trademarks of International Business Machines Corporation.

Dow Jones News/Retrieval is a registered service mark of Dow Jones & Company, Inc.

DUMP.EXE is a trademark of Phoenix Technologies Ltd.

EPSON is a registered trademark of Epson Corporation.

Epson Equity is a trademark of Epson America, Inc.

FLASH is a trademark of Software Masters, Inc.

Hercules Graphics Card is a trademark of Hercules Computer Technology.

Keyworks is a trademark of ALPHA Software Corporation.

GW—BASIC, Microsoft Excel, Microsoft QuickBASIC, Microsoft Windows/386, Microsoft Windows Development Kit, Microsoft Windows Write, Microsoft Word, MS-DOS, and XENIX are registered trademarks of Microsoft Corporation.

Mouse Systems is a trademark of Mouse Systems Corporation.

NU and the Norton Utilities are registered trademarks of Peter Norton Computing.

Novell is a registered trademark of Novell, Inc.

PageMaker is a registered trademark of Aldus Corporation.

Paradox and Query By Example are registered trademarks of Borland/Ansa Software.

PC Paintbrush is a registered trademark of ZSoft Corporation.

Post-It is a registered trademark of 3-M Corporation.

ProKey is a trademark of RoseSoft, Inc.

RamQuest 50/60 is a trademark of Orchid Technology, Inc.

Reflex: The Analyst is a trademark of Borland/Analytica, Inc.

Seagate is a registered trademark of Seagate Technology.

ScanJet is a trademark of Hewlett-Packard Co.

SDACHMD is the copyright of SDA Associates.

The Source is a service mark of Source Telecomputing Corporation, a subsidiary of the Reader's Digest Association, Inc.

Tandy is a registered trademark of Tandy Corporation.

Turbo Lightning is a trademark, and Quattro, SideKick, SideKick Plus, and SuperKey are registered trademarks of Borland International, Inc.

Ventura Publisher is a registered trademark of Ventura Software, Inc.

WordPerfect is a registered trademark of WordPerfect Corporation.

WordStar 2000, WordStar 2000 Plus, WordStar Release 3, and WordStar Release 4 are registered trademarks of MicroPro International Corporation.

UNIX is a trademark of AT&T.

Introduction

Since its introduction in 1981, MS-DOS has grown to be the most widely used operating system in the world. Thousands of applications programs have been written for the MS-DOS operating system. With over 10 million users world-wide, MS-DOS affects more people than any software product ever written.

As MS-DOS has evolved, Que Corporation has helped hundreds of thousands of personal computer users get the most from MS-DOS. *MS-DOS User's Guide*, Special Edition, represents Que's continuing commitment to provide the best microcomputer books in the industry. *MS-DOS User's Guide*, Special Edition, is a comprehensive learning tool and reference volume for users of MS-DOS and PC DOS. This Special Edition reflects the maturity of MS-DOS and the far-reaching impact that MS-DOS-based microcomputing has had on people and the way they work.

MS-DOS is the operating system of choice for the majority of personal computer users and will remain so for the foreseeable future. *MS-DOS User's Guide*, Special Edition, offers DOS users a source of information to organize their work with the PC more effectively, make their hardware respond more efficiently, and answer those questions that *must* be answered.

Who Should Read This Book?

MS-DOS User's Guide, Special Edition, is written and organized to meet the needs of a large group of readers. The book is especially well suited for readers who are proficient with applications programs, but who could benefit from a better understanding of DOS. *MS-DOS User's Guide*, Special Edition, is also for those people who "get by" at the operating system level, or those who have grown beyond elementary tutorials.

Perhaps you have upgraded your hardware to a more powerful PC with more memory and disk capacity, or have upgraded your version of MS-DOS and want to take advantage of its new or expanded features. You may be the informal

"expert" in a group of PC users and need to assist other users or answer their DOS questions. Maybe you are just beginning to use a PC, but you know that you're a fast study and would like to get up to full speed quickly. If you find that you fit into any of these categories, the comprehensive *MS-DOS User's Guide*, Special Edition, is a "must have" volume.

What Hardware Is Needed?

This book applies to the family of IBM personal computers and their close compatibles. This list includes the IBM Personal Computer, the IBM Personal Computer XT, the IBM Personal Computer AT, and the IBM PS/2 series. Dozens of other computers which use MS-DOS, and are manufactured or sold by COMPAQ, Zenith Data Systems, EPSON, Leading Edge, Tandy, Dell, Northgate, CompuAdd, and others are also covered by this book. If your computer runs the MS-DOS operating system, you have the necessary hardware.

What Versions of DOS Are Covered?

This book specifically covers MS-DOS and PC DOS versions 3 and later. Special attention is given to versions 3.3 and 4. Users of DOS V2 will also find their version covered to their satisfaction. If you use DOS V2, you can generalize much of the book to that version, but many useful DOS features and commands included in the newer DOS versions are not available to you.

What Is Not Covered?

A book that covers all aspects of MS-DOS would be a weighty book indeed. *MS-DOS User's Guide*, Special Edition, does not include the DEBUG or LINK commands, nor does it include a technical reference to the applications programming interface that DOS provides programmers. If you are interested in programming at the operating system level, you will find that Que offers a complete line of books that cover DOS programming.

Also not included in this book are computer-specific setup or configuration commands, such as IBM's SETUP for the PS/2 and Toshiba's CHAD for laptop displays. Although these commands are often distributed with the same disks as MS-DOS, they are too variable to be covered adequately here. Your computer-supplied manuals and your PC dealer are the best sources of information about these machine-specific features.

The Details of This Book

You can flip quickly through this book to get a feeling for its organization. You will find that *MS-DOS User's Guide*, Special Edition, approaches DOS in a logical, functionally defined way. The material in this book is arranged into six main parts and a set of appendixes.

Part I—Understanding the Fundamentals of DOS

Part I is devoted to explaining the fundamental role of DOS in a working PC. The approach taken is non-programmatic, but insightful. The main blocks of DOS are discussed in individual chapters.

Chapter 1, "Understanding the Role of DOS," looks at the common uses of the term DOS and builds a definition of the layered PC. This chapter demonstrates that DOS is not a single software entity, but a set of modules, each specializing at its own level in a PC.

Chapter 2, "Using the DOS File System," is a tutorial on DOS's extensive provisions for using disks. An entire range of topics is covered—from why disk drives are the primary storage media to how DOS tracks your files in a multidisk filing system. This chapter fills the knowledge gap about disks and files that texts aimed at beginning users often leave out.

Chapter 3, "Understanding the Command Processor," explains how DOS bootstraps itself and the PC into the tool you use to do your computer work. The chapter covers the general rules and considerations for all DOS commands as well as the special keyboard keys. The commands are categorized and described in several useful tables.

Chapter 4, "Gaining Control of the Hardware," looks at the nature of device drivers and their role in DOS. Important considerations—such as how programs operate and how memory is divided—are covered in this chapter. System configuration concepts are explained through a discussion of device drivers. The concept of redirection and piping is introduced within the context of DOS's device-independent commands.

Part II—Working with the Core Commands

The core commands of DOS are presented in functional chapter blocks in Part II. The commands presented in Part II are those most commonly used. Familiar commands are explained in a way that gives you new insight into the commands. Commands that you are less likely to use are treated in a slightly more tutorial way. With this approach, you can use Part II to get the most from what you know—and expand your base of daily DOS commands to other useful utility commands.

Chapter 5, "Preparing and Maintaining Disks," concentrates on disk-level commands. Partitioning, formatting, disk copying, and disk comparison are covered as well as other disk-level utilities. Examples and exercises help you understand these disk-level commands.

Chapter 6, "Managing and Navigating the Directory Hierarchy," reviews the important hierarchical directory file-management system. The commands presented in this chapter enable you to manage DOS directories in an effective fashion. Examples help show the way.

Chapter 7, "Managing Your Files," illuminates the file-level DOS commands. Because you are likely to spend most of your time with DOS working with files, this chapter offers an in-depth view of the file-level commands. Each command includes examples that will help you appreciate the full power of these important commands.

Chapter 8, "Working With System Information," covers the commands that set and retrieve system information in your DOS-based computer. These commands are often neglected, but their use keys you into the "control panel" of DOS. Whether you oversee a single PC, or help other users with their PCs, you will find these commands useful.

Chapter 9, "Gaining Control of Devices," explains the DOS commands that control the behavior of logical DOS devices. By using these commands, you can control the way DOS sees your system's drives and directories. You will learn how to use your printer while doing other computer work, and you will see how to use DOS pipes and filters effectively.

Part III—Expanding Your Use of DOS

Part III provides the information you need to tap the expanded power available with batch file processing and screen and keyboard control. Creating and editing your own batch files is a sure way to become a more proficient DOS user—these chapters show you how.

Chapter 10, "Using EDLIN," provides a tutorial approach to DOS's built-in text-file editor. The examples developed in this chapter show you how to use EDLIN as a day-to-day utility. With the careful attention given to EDLIN's practical use, you will learn the skills needed to quickly compose a text file. Practical examples of using EDLIN to create memos and batch files are also presented.

Chapter 11, "Understanding Batch File Basics," guides you through the process of creating batch files. The commands related to batch files are explained in a tutorial style. Useful examples help make mastering the basics of batch files easier.

Chapter 12, "Understanding ANSI.SYS," shows you how to make the "plain vanilla" DOS screens look colorful and controlled. The details of the ANSI.SYS driver are presented in workshop fashion. You will learn how to reassign keys, control the cursor's position on the screen, display the date and time, and more.

Chapter 13, "Learning Advanced Batch File Techniques," builds on what you learned in Chapters 10, 11, and 12. You learn how to automate utility DOS work in a batch file environment that includes the colorful ANSI.SYS screen driver capabilities. The batch files presented in this chapter are useful on their own, but they also serve as useful models for designing your own advanced batch files. Whether you are customizing your own DOS work, or providing an easy-to-use set of utilities for a DOS beginner, this chapter shows you how.

Part IV—Advancing Your DOS Capabilities

Part IV sheds light on the often overlooked or misunderstood aspects of utilizing the full capabilities of DOS.

Chapter 14, "Configuring Your Computer," is a comprehensive collection of DOS commands and directives that are explained to help you get the best performance from your PC. You learn several useful techniques—from creating a CONFIG.SYS file to using disk-caching software with extended memory.

Chapter 15, "Understanding the International Features of DOS," steps you through the complicated, but sometimes necessary, configuration of a PC to various international language standards. In today's "global village," you may find the need to "go international" with your PC. This chapter provides tutorial and reference material that can make the job easier for you.

Chapter 16, "Using the DOS Shell," directs the configuration and use of the graphical user interface included with DOS V4. You will see the features and benefits of the DOS Shell.

Part V—Command Reference

Part V contains the comprehensive Command Reference. The commands, which are arranged in alphabetical order, are shown with syntax, applicable rules, examples, and possible screen messages. When a command is also covered elsewhere in the book, a chapter reference is provided. You can use the section both as a reference when you have problems and as a source of practical advice and tips during "browing sessions." In all, the Command Reference is a complete, easy to use, quickly accessed resource on the proper use of DOS commands.

Appendixes

The *MS-DOS User's Guide*, Special Edition, also includes six appendixes containing useful summary information. Appendix A lists the changes and additions of commands between version changes. Appendix B is an ASCII code chart.

Appendix C shows the DOS control and editing keys. Appendix D outlines the preparation of a hard disk. Appendix F provides the ANSI control sequences supported by DOS through ANSI.SYS. Appendix G lists character tables of the standard code pages.

Learning More About DOS

If *MS-DOS User's Guide*, Special Edition, whets your appetite for more information about DOS, you are in good company. Thousands of *MS-DOS User's Guide* readers have gone on to purchase one or more additional Que books about DOS.

Although *MS-DOS User's Guide*, Special Edition, is a comprehensive guide to DOS, no one book can fill all of your DOS and personal computer needs. Que Corporation publishes a full line of microcomputer books that complement this book.

You are probably using DOS on a PC equipped with a hard disk drive. The key to efficient computer use is effective hard disk management. Que's *Managing Your Hard Disk*, 2nd Edition, provides you with the in-depth information you need to get the most from your hard disk.

Newer versions of DOS support powerful equipment to help your programs run quickly and efficiently. If your current hardware is not up to the task of running the extensive new software packages, examine Que's *Upgrading and Repairing PCs*. This informative text shows you how to get the most from your hardware and how to upgrade your system to handle the new breed of high-powered software, such as Lotus 1-2-3 Release 3, dBASE IV, Windows, and WordPerfect. According to Mark Brownstein of *InfoWorld*, *Upgrading and Repairing PCs* is "one of the best books about the workings of personal computers I've ever seen; it will be a useful, easy-to-read, and interesting addition to most anyone's library."

If you are a corporate trainer or need to teach novices about DOS, you may be interested in purchasing Que's *MS-DOS QuickStart*. This book features a graphics approach to DOS and is geared to the beginner. Que also offers the *DOS Workbook and Disk*—an interactive learning tool for the classroom environment or motivated self-starters.

Other reference tools for the DOS user include *DOS Programmer's Reference*, 2nd Edition, *DOS Tips, Tricks, and Traps*, 2nd Edition, and *DOS QueCards*. The *DOS Programmer's Reference* offers comprehensive coverage of the programming aspects of MS-DOS. Que's *DOS QueCards* is a desktop reference tool that DOS users of any experience level will find invaluable. *DOS Tips, Tricks, and Traps*, 2nd Edition, provides a topical collection of in-depth DOS tidbits.

All of these books can be found in better bookstores worldwide. In the United States, you can call Que at 1-800-428-5331 to order books or obtain further information.

MS-DOS User's Guide, Special Edition, follows the Que tradition of providing quality text that is targeted appropriately for the DOS user. Because of the dedication to this goal, Que ultimately has only one way of getting better—by hearing from you. Let Que know how you feel about this book or any other Que title. Que wants to keep improving its books, and you are the best source of information.

Conventions Used in This Book

Certain conventions are followed in this edition to help you more easily understand the discussions. The conventions are explained again at appropriate places in the book.

Uppercase letters are used to distinguish file names and DOS commands. In most cases, the keys on the keyboard are represented as they appear on your keyboard, with the exception of the arrow keys. Instead of the arrow symbols, the corresponding terms, such as "up arrow" or "up-arrow key," have been used.

Ctrl-Break indicates that you press the Ctrl key and hold it down while you press also the Break key. Other hyphenated key combinations, such as Ctrl-Z or Alt-F1, are performed in the same manner.

Special words or phrases that are defined for the first time appear in *italic* characters. Words or phrases you are to type appear in **boldface** characters.

All screen displays and on-screen messages appear in the following special typeface:

```
This is a screen message
```

The system prompt (`A>`, `C>`, etc.), which appears on-screen, also appears in the special typeface.

The notation for issuing commands and running programs appears, in fullest form, in lines like the following:

*dc:pathc***COMMAND** *filename.ext /switches*

Different conventions are used to distinguish between optional and mandatory parts of a command. Items displayed in **boldface** are mandatory; you must always give this part of a command. In this example, to issue the command called **COMMAND**, you must type the word **COMMAND** as it appears.

For many mandatory items, you must substitute the appropriate information, such as a file name, a device name, a drive name, or a path name. Note the following syntax line:

ASSIGN d1=d2 ...

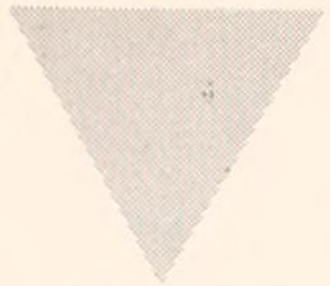
You must substitute the appropriate drive letters for **d1** and **d2**. In this case, **d1** and **d2** are variable, but they are also mandatory. Remember that all mandatory items on a command line must be provided in order for the command to perform the operation described.

Items in *italic characters* are optional; you supply these items only when needed. For example, you substitute the appropriate disk drive and path name for *dc:pathc*\. You substitute for *filename.ext.* a file name or other item. You use the appropriate character or characters for */switches*.

In most cases, the text will also help you distinguish between optional and mandatory items, as well as those mandatory items that are variable.

Uppercase letters are usually used in the examples, but you can type commands in either upper- or lowercase letters. For a few commands, the case of the letters makes a difference. When the case is significant, the difference is specifically mentioned in the discussion of the command.

Part I



Understanding the Fundamentals of DOS

Understanding the Role of DOS
Understanding the DOS File System
Understanding the Command Processor
Gaining Control of Hardware

1

Understanding the Role of DOS

Just a few years ago, most people never got to see a real computer, let alone use one. Computers were large, expensive, and rare. Most people who worked in a computerized environment had no real knowledge of how the machine worked. Keypunch operators entered source documents into their machines to produce cards. These punch cards were then fed into the mysterious mainframe computer by a card reader. The mainframe used the data from the cards to do math-intensive calculations whose results became output on green-bar paper. Although paper was a common medium for output, screen displays were rare and expensive.

Science fiction movies, such as *2001: A Space Odyssey*, depicted the computers of the future as semihuman thinking machines, complete with columns and rows of blinking lights, input devices that looked like crystal cylinders, and output in the form of a man's monosyllabic voice. Perhaps the few individuals that understood how to create those early programs knew that computers would become more practical someday. They saw that computers could perform in a few minutes the practical work that took humans a few days to do.

In those early days, the prediction for the computer was that of increasing complexity and mystery. The notion that everyday people in everyday jobs would have access to their own personal computers (and know how to use them) was strictly "pie in the sky." There was as much belief that people would abandon commercial aviation and begin flying their own private airplanes. The expense and expertise required seemed overwhelming.

Despite the fear of crystal input devices and banks of blinking lights, ordinary people continued to move closer to the computer. Video display terminals replaced punch cards and green-bar paper. Screen messages in "plain English" replaced cryptic numbers (or at least most of them). Minicomputers appeared where mainframes had been supreme. In the computer's mysterious insides,

thousands of transistors were being replaced by dozens of integrated circuits, which were being replaced in turn by one small microprocessor and a handful of support-circuit components. Microcomputer hobby shops opened next to model airplane shops. The microcomputer industry was born.

Cue:
The MITS Altair introduced in early 1975 was the first widely available microcomputer. (MITS is an acronym for Micro Instrumentation Telemetry Systems.)

In the past few years, the microcomputer industry has grown from a cottage industry to a multibillion dollar business. The idea of an individual computer for each user has moved from a "pie-in-the-sky" idea to reality. You now live in the era of the personal computer. Much of the credit for this transition is due to International Business Machines, Inc. (IBM), for designing and marketing the IBM Personal Computer, and to Microsoft, Inc., for producing operating system software for the IBM PC. These companies benefited from the growing wave of acceptance for personal computers (also called PCs) that had begun with the CP/M operating system and the Apple Computer. In this chapter, you will get a clearer view of the world's most popular operating system, MS-DOS, as well as its role in the personal computers (both IBM's and others like them) that you use today. With this clearer view, you will be better equipped to understand how to control and manage PCs using the MS-DOS operating system.

What is DOS?

Cue:
DOS is an acronym for Disk Operating System.

DOS is an acronym for *D*isk *O*perating *S*ystem. Nearly every computer has some sort of disk operating system. Those that do not are severely limited in reliable data storage. Disk operating systems manage many of the technical details of computers as well as disk-file storage. Disk management, however, is perhaps the most important service these operating systems provide.

Many computers, including personal computers and large, multiuser computers, use these three letters—DOS—as part of their operating system's name. Microsoft's MS-DOS and IBM's version of MS-DOS, PC DOS, are examples. So are Apple's DOS 3.3 and ProDOS.

Cue:
Various other operating systems are called DOS.

These various operating systems are all called DOS. But if numbers count, more people associate the term DOS with MS-DOS than with any other disk operating system in the world. There are millions of MS-DOS users. When you read the term DOS in this book, you will be reading about MS-DOS. MS-DOS is a single-user, single-tasking disk operating system that provides the framework of operation for millions of today's personal computers.

The Variations of MS-DOS

Cue:
Variations of MS-DOS are very close in design.

MS-DOS is an operating system that accommodates variation in its exact makeup while retaining its core characteristics. This capability for variation enables various computer manufacturers to adapt MS-DOS to their computers.

Both IBM PC DOS and COMPAQ DOS, for example, are variations of MS-DOS. Today, all commonly available variations of MS-DOS are very close in design and operation. There is a good reason for this closeness. MS-DOS works with computers designed around Intel's 8086 microprocessor family. This family includes the 8088, 8086, 80286, 80386, and 80486 microprocessors. The enormity of the original IBM PC's influence on the PC market convinced other PC manufacturers using the 8086 family to follow closely IBM's personal computer design. Because IBM used an open approach to designing the IBM PC, other companies were able to configure their computers to use programs and hardware designed for the IBM PC. From the point of view of DOS, these various computers are virtually alike, even though the PCs may be more advanced than the original IBM PC.

The closeness of its design to the IBM PC determines to what degree a PC is said to be *IBM-compatible*. The buying public has demanded a high degree of IBM compatibility in PCs. Most PCs (and their respective variations of MS-DOS) that were not designed to be highly compatible with the IBM PC have either been discontinued, sell in specialty markets, or are available in countries other than the United States. Because of this "shaking-out" of the market, most compatible PCs successfully operate using a version of IBM's PC DOS even though most compatible manufacturers usually offer their own variation of MS-DOS.

The variations in MS-DOS implementations among PC manufacturers are seen by users as subtle differences, usually limited to the names of a few commands or the way parameters are given. Some manufacturers include in their MS-DOS packages additional utilities that work with a feature of that manufacturer's model of PC. Yet, as a general rule, if you are proficient at using one variation of MS-DOS, you are proficient with all variations of MS-DOS. In a practical sense, the terms MS-DOS, PC DOS, and DOS are interchangeable.

Cue:
DOS has evolved through four major versions.

Another difference involves the evolution of the product itself. As MS-DOS has evolved, the core product has been enhanced several times. Each release of enhancements is a distinct version of the program. Since its appearance in the summer of 1981, DOS has evolved through four major versions. (Changes between versions are summarized in Appendix A.) Even with the introduction of Microsoft's OS/2 and the presence of several versions of the UNIX operating system, DOS remains strong. You can expect your investment in learning DOS to continue paying dividends in any future computing. Most industry experts predict that DOS will have a presence in the PC picture for years to come. Even if you move to OS/2 or UNIX at some point, you will find that much of your DOS expertise can be used with these operating systems.

Understanding DOS in Context

Depending on the context of activity and use, the exact meaning of the term DOS can vary quite a bit. Most users read about DOS and talk about DOS in ways that are familiar to them.

Consider the PC buyer who asks the dealer, "Does this computer come with DOS?" If the dealer's answer is "Yes," that affirmation doesn't necessarily mean that the buyer will receive a set of master DOS disks and manuals in a bound package, even if that is what the buyer had in mind. To the buyer, the term DOS means the DOS *package*. The dealer, on the other hand, may answer "Yes" because the hard disk on the PC includes MS-DOS V2.2 and utility files already loaded. To the dealer, "Yes" is a fair answer even though manuals and master diskettes aren't part of the deal. Both buyer and dealer are using the term DOS in different contexts.

Likewise, an applications software package may include a help screen that flashes the message `Press F10 to return to DOS`. The message doesn't mean, "Return to the DOS manuals and master disks." This message is the software's way of telling you how to finish your work with the package.

Because its meaning can change based on the context, agreeing on a common definition is easier when everyone concerned knows the context in which the term DOS is being discussed. The next sections look at some of the common contexts of the term DOS. As you look at each context, you will be building a definition of the term DOS.

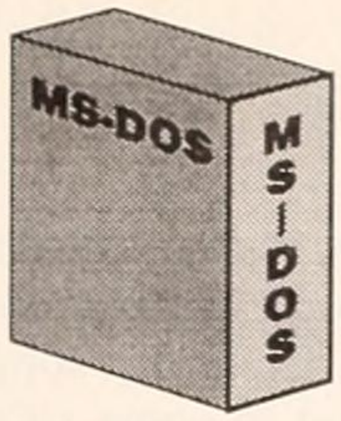
Figure 1.1 explains the use of the term DOS in different contexts. Each picture can represent the term DOS in the right context. As the meanings progress from top to bottom, the meaning of DOS becomes less literal and more abstract. You can touch and see a DOS package, but you must conceptualize the idea of DOS working in a computer.

The DOS Package

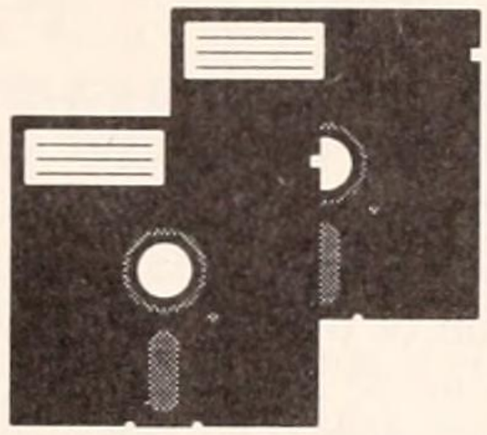
The buyer's question to the dealer, "Does this PC come with DOS?" usually means "Does it come with a DOS package?" This is a product-oriented meaning for the term DOS. Dealers and manufacturers usually supply or sell DOS as a stand-alone package or as an accessory package to a PC. Most often the DOS package includes one or more manuals and two or more disks. The DOS software itself is on the disk(s).

Cue:
Some DOS packages include supplemental programs.

PC DOS V4.0, for example, comes packaged with 2 or 5 disks, depending on the model of computer intended for the software. PC DOS V3.3 comes in a package with 1 or 2 disks. COMPAQ supplies DOS V4.0 with 2 DOS disks and 1 supplemental disk. DOS packages supplied by other computer manufacturers may



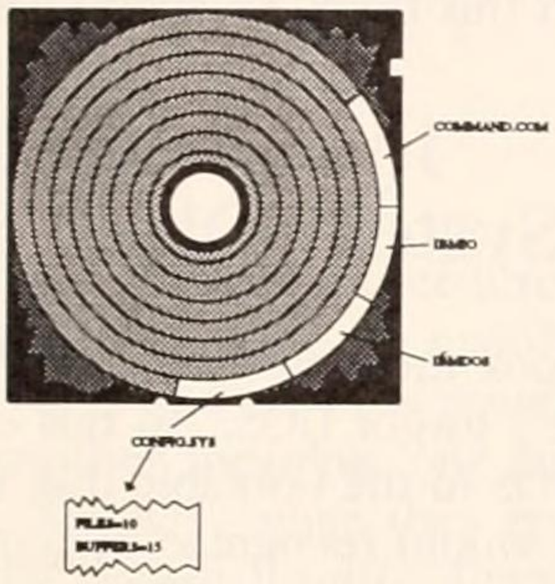
The DOS package



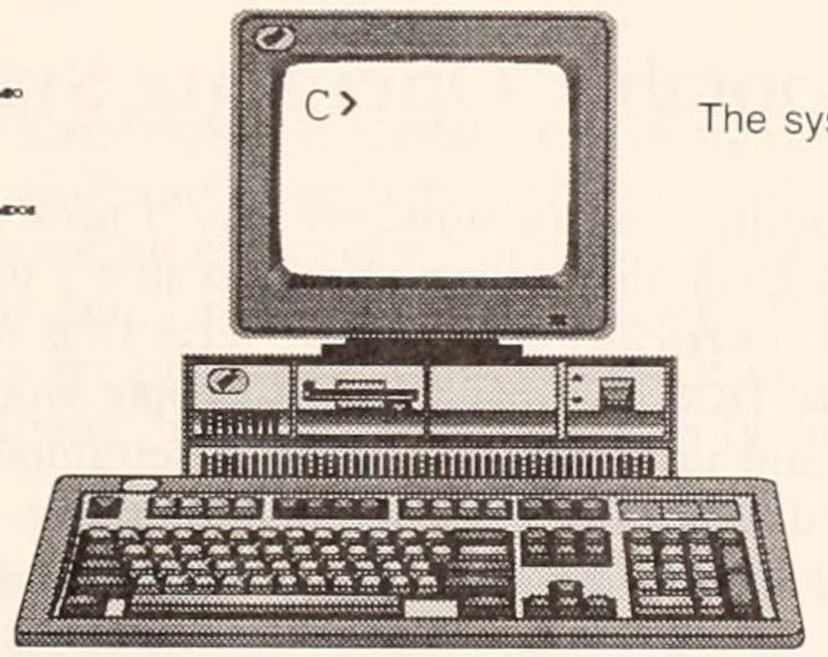
The DOS system disks

Fig. 1.1

The term DOS in context.



The hidden core of DOS



The system prompt

have more or fewer disks. The disks may contain supplemental programs for the PC such as GW—BASIC and MS-LINK. These supplemental programs are often considered part of DOS because the manufacturer included them in the physical DOS package. Depending on the context, supplemental programs may be associated with the term *DOS package*, but for the purpose of this book, you should consider such supplemental programs to be outside of DOS.

Normally, the DOS package contains a DOS reference manual. The manual is an important supplement because it contains specific information about your version of DOS. Many PC users, however, prefer to use another book, such as this book or other books about DOS published by Que Corporation, for their daily DOS reference. It's not surprising that DOS users may prefer another, perhaps more specialized, book to the package's DOS manual. DOS users can select a book that fits their needs closely, whether they are beginning, intermediate, or

advanced DOS users. On the other hand, the DOS reference manual included in the DOS package must attempt to supply an accurate description of that version of DOS to novice and advanced users alike.

Reminder: *Keep any specialty manuals at hand for reference.*

Many manufacturers include in the package other manuals that look similar to the DOS reference manual and contain information specific to the computer model or its auxiliary software. COMPAQ, for example, supplies a manual describing the operation of the tape unit included in some COMPAQ models. Although the discussion of the tape unit includes references to DOS, this manual is not considered to be a standard part of DOS for all PCs. The topics in these specialty manuals are not covered in this book, so keep those additional manuals at hand for reference.

The Bootable Operating System Disk

Perhaps you have heard someone say, "I took the laptop home to work on the report, but I couldn't do anything because I forgot DOS." In this case, the laptop user is generically applying the term DOS to the bootable disk that contains the external DOS commands. Most people would recognize the context of this comment and understand that the absentminded user had forgotten the DOS disk needed to boot the floppy-only computer. In all likelihood, the user's office PC was equipped with a hard disk and didn't need a floppy to boot.

Cue: *The DOS system files contain computer-level instructions.*

DOS disks, like the DOS package, are physical products. What makes the disks special is that they contain DOS files. These DOS files contain the computer-level instructions that provide the functional aspects of DOS. The PC doesn't have these computer-level functions built in. The instructions are logical—not physical. Because a physical definition is incomplete, the role of the instructions and the operation must be viewed conceptually.

This conceptual view includes the steps that make the computer "compute." Each time it is turned on, the PC must refresh itself by reading in the DOS instructions from the DOS system files on the PC's start-up disk to the PC's working memory. The PC is then ready to run your favorite software.

All disks must be prepared by DOS before they can be used by DOS. Only disks prepared with the DOS system files can start your PC. Disks that include the DOS system files are the DOS boot disks. In fact, many people refer to a bootable disk as being DOS.

Caution: *Don't use the master DOS disks for everyday work.*

If you are fairly new to DOS, here's a word of wisdom: Don't use the master DOS disks that came in your DOS package for everyday work. The manufacturers intended for you to make copies of these master DOS disks; the copies can then be used when you need to use the DOS disks. Be sure to store your masters in a safe place. You should have at least one known good (tested) set of working DOS floppies even if your computer has a hard disk that contains the DOS files.

With versions of DOS prior to V4.0, you can use the DOS command DISKCOPY to duplicate and prepare working copies of the packaged DOS disks. With DOS V4.0 and later versions, you can use the Select program to install DOS both on a hard disk and a floppy-based copy. The Select program automatically runs when you start your PC with the package's Install disk. The Install program eliminates the need to use DISKCOPY by guiding you through the creation of DOS working disks.

Even if DOS is installed on your hard disk, do not be without floppy-based copies of your DOS package disks. You may need them if your hardware experiences problems or some software holds your hard disk hostage.

The System Prompt

Cue: *COMMAND.COM is the most visible part of DOS.*

A certain company's weekly procedure book includes the following instructions for downloading a file from the mainframe computer: "When the file is downloaded, exit to DOS and start the translation program." In this context, the term DOS has yet another meaning, one familiar to most PC operators. Many users think they are "in DOS" when they are presented with the DOS prompt, sometimes called the *system prompt*. Being "in DOS" is the same as being "at the operating system level."

Many applications programs provide messages that tell you they are `Returning to DOS`, `Exiting after a fatal error to DOS`, or `Temporarily invoking DOS`. These messages mean that the program is turning control over to COMMAND.COM, which then displays the system prompt on-screen. You recognize the system prompt as `A>` or `C>` or perhaps other text followed by the characteristic `>` symbol. COMMAND.COM is the DOS program that processes the commands you issue. Because DOS commands are issued at COMMAND.COM's system prompt, and issuing commands is the way most users see DOS in operation, it's easy to see why people frequently use the term DOS to mean an interaction with COMMAND.COM.

COMMAND.COM is a very important part of DOS. Perhaps PC users think of COMMAND.COM's operation as being the essence of DOS because it is so visible. Or perhaps users equate issuing commands with performing DOS-level PC-management work. Chapter 3 discusses COMMAND.COM more fully, and the second part of this book is devoted to the DOS commands users most often issue. But issuing commands is only a part of using DOS. Many of the commands you issue at the DOS prompt work with the file system. Chapter 2 provides an inside view of the file system that makes using disk- and file-related DOS commands more meaningful.

The Hidden Core of DOS

Cue:
Programmers can access the internal operations of DOS.

An instructor of C-language programming for the PC tells his students: "You can use the INT86() function to get directly at DOS." In the context of the instructor's statement, DOS is the collection of services that DOS offers in the form of built-in groups of related instructions or software routines. These routines provide the core of the services DOS provides to your applications programs. The extensive file-system management provided by DOS, for example, is made up of these routines. Programmers can access these software routines to perform a variety of internal operations with the PC. These functions are common and repetitive actions that operating system designers have included in DOS to make life easier (and programs more uniform) for those people who write programs for PCs. DOS commands themselves rely on these service routines to do standard low-level computer work. Most computer languages provide a built-in access to DOS service routines so that a programmer (or a program) doesn't see the details of how DOS works with the PC. With the interface to DOS service built-in, computer languages, and the programs they produce, use DOS uniformly.

The programming aspects of DOS are often as invisible to DOS users as interaction with COMMAND.COM is obvious. Yet, if you ask an assembly-language programmer to tell you about DOS, you will most likely hear about interrupts, BIOS services, registers, file handles, returns, and other aspects of working with the core of DOS.

Fortunately, you do not have to know about these programming concerns to operate your PC. Having some idea of what goes on at the program level, however, may give you more insight into how a DOS command or your word processor works. Programming at the DOS level is fascinating and rewarding to many, but this book will not attempt to teach you the programming aspects of DOS. If you want to know more about programming at the DOS level for additional insight or to write actual programs, you can refer to one of the books published by Que Corporation on this topic. *DOS Programmer's Reference*, 2nd Edition, by Terry Dettmann, and *Using Assembly Language*, by Allen Wyatt, are suggested.

If the programming aspects of DOS do not interest you, do not feel alone. Most DOS users are generally uninformed about "what goes on in there," and they do just fine. As your DOS skills increase, however, you may find that having a general layman's notion of "what goes on in there" will enable you to become a more self-reliant PC user. Chapter 4 offers a general overview of this invisible but essential part of DOS.

The Configuration Capability of DOS

The error-correction instructions in the spreadsheet manual say, "If the message `Too many open files` appears, you may need to make adjustments in DOS." In this example, the spreadsheet-software manual is using the term DOS to mean "the configuration aspects of DOS." Although most computers that use MS-DOS as an operating system are highly compatible with each other, individual aspects of each PC's hardware and software components can vary.

Cue:
The flexibility of DOS enables you to tune your PC to your needs.

In nearly all respects, this variation is healthy for PC users. You don't, for example, have to deploy memory for 20 open files when you will never need more than 10. Some operating systems make configuration decisions for you. Fortunately, DOS is flexible, and many of the parameters of DOS are under your control.

If you need only 10 open files, the flexibility of DOS enables you to tune your system to your needs. The configuration aspects of DOS also keep you from being locked into a rigid PC design. You do not have to use a parallel printer when your serial printer serves your purposes just as well. As you will see in Chapters 14 and 15, the configuration aspects of DOS include many variables, ranging from the amount of memory in your PC to the country's character set you use.

Many DOS users do not take advantage of the configuration capabilities of DOS because someone else (usually the dealer) set up their PC for them. Others do not take advantage because their PC seems to work well as it is. As software becomes more complex and hardware continues to make technical strides, however, your ability to use these configuration capabilities will become increasingly important as you fine-tune your PC for maximum efficiency. Part IV of this book explains the use of the configuration commands.

Examining the Layers of DOS

Actually, the contextual meanings of DOS help to point out that DOS isn't a single, seamless software entity. DOS is designed in modular parts. The parts of DOS cooperate with other software and hardware parts to produce the whole effect that you know as computing. Between the fundamental electrical signals that make your hardware function and the user-friendly screens of your applications programs, the parts divide into layers of responsibility. No one layer is totally responsible for the job of computing—not your program, not DOS, and not the hardware that sits on your desk.

As a PC user, you see the surface layer of your PC as the program that prepares a memo or budget spreadsheet. When you are at the system prompt or using the DOS Shell—you can get the feeling that you are in a lower layer of the PC where you manipulate directories and files.

Occasionally, you catch a glimpse of lower layers of responsibility. In these glimpses, you see evidence of a more hidden layer of DOS. You see system information or error messages reporting lost clusters in a file allocation table or memory parity errors. These messages originate from hidden layers of the system that DOS passes to the surface for you to see. The hidden layers are responsible for the many fundamental tasks that comprise the low-level support system of your computer.

The DOS layers work on the surface with you and your programs as well as with a layer of low-level support routines permanently programmed into your PC. To understand the concept of layers and how DOS fits in, start by thinking of the hardware as being the core or innermost level of the working PC system. The diagram in figure 1.2 will help you follow the levels outward.

Fig. 1.2

The modular layers of a working PC.

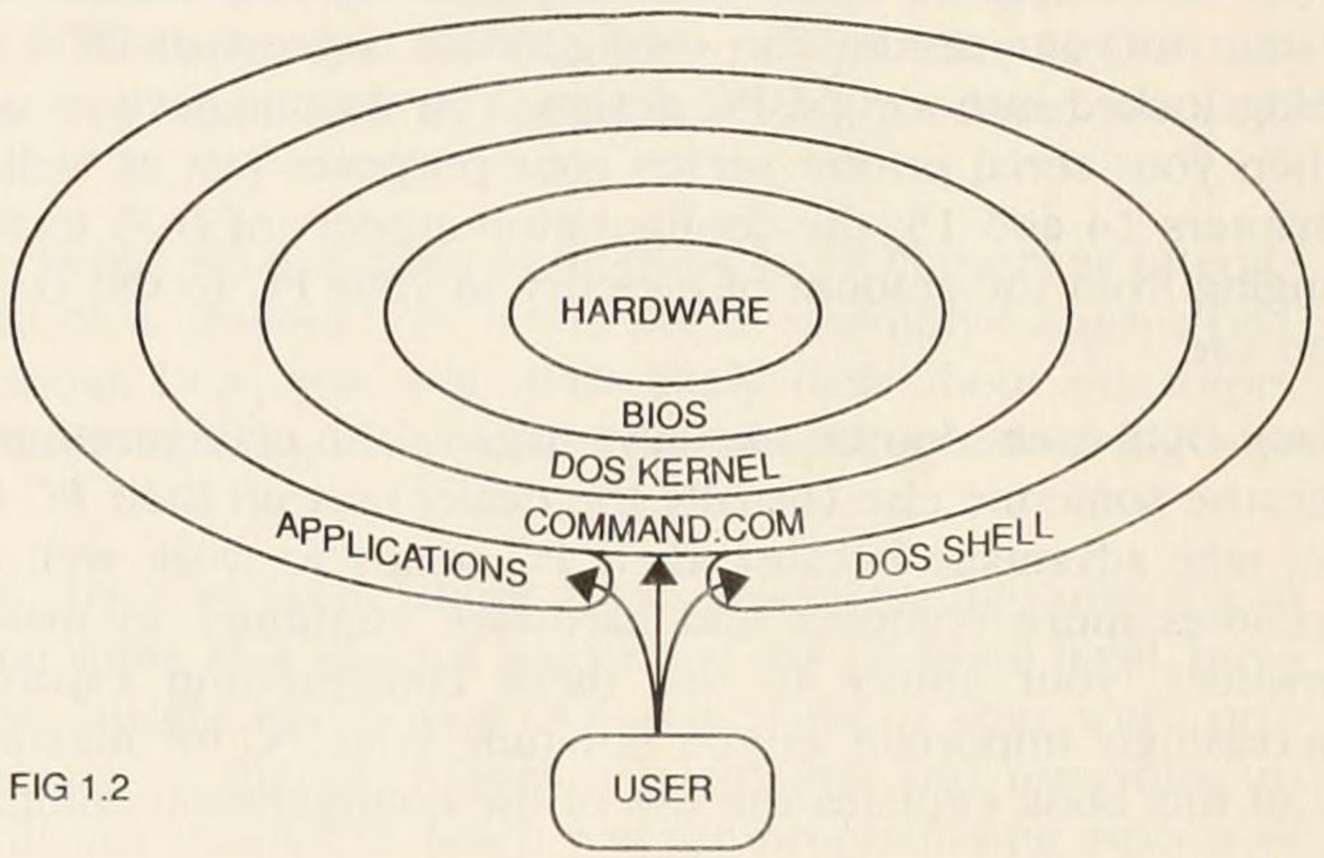


FIG 1.2

When IBM designed the IBM PC, hardware designers integrated other manufacturers' hardware components into a personal computer design specification. This hardware design was the core of a working computer. The core of the circle in figure 1.2 represents the hardware as the core of a working PC.

With a core hardware design specified, IBM designed the low-level BIOS (Basic Input/Output System) software routines (see fig. 1.2). The BIOS routines handle the specifics of the hardware at a detailed level. An operation as seemingly simple as translating a key press from the keyboard is a complicated elemental task. The BIOS is designed to be responsible for carrying out this complicated task and many others. Because the BIOS must work directly with the PC's hardware, the BIOS is built directly upon the hardware's requirements and capabilities. The BIOS is modular because it can be adapted to changing hardware designs

and still provide the same elemental task. The BIOS is a modular layer of a working computer that connects to the hardware and acts as a basis for the DOS kernel layer above it.

The next level of the circle is the DOS kernel, which is the core of the DOS software. In a working PC, the kernel is insulated from the hardware by the BIOS. The kernel relies on the elemental services of the BIOS layer to access the features of the hardware. The kernel is modular because it can evolve into higher-level services such as a file system and device redirection while the BIOS, on which the kernel is built, remains relatively unchanged. Figure 1.2 shows the DOS kernel layer connecting to the BIOS layer and providing a foundation for COMMAND.COM.

Cue:
The DOS kernel is the core of the DOS software.

COMMAND.COM, the DOS command processor, acts as a module of DOS. COMMAND.COM is best known for providing a user-interactive interface to commands. Through COMMAND.COM, a user sees the PC as a configurable, maintainable computer. DOS commands use DOS kernel services to enable you to do "housekeeping" on your PC. COMMAND.COM is a modular layer of a working PC because COMMAND.COM can be replaced by a user-provided alternate command processor. Figure 1.2 shows the COMMAND.COM layer of a working computer built upon the DOS kernel.

Your applications programs, as well as the DOS Shell (which is also an application), make up the outermost layer of the working computer (see fig. 1.2). When you are running your favorite program, you most likely don't think about all of the layers of the working PC that are supporting that program. The layers are there, however—each layer playing an important role in your program's operation.

The Hardware

A PC's hardware is a collection of electronic and electro-mechanical components. Some of these components operate at the system level; others operate at the peripheral level. System-level components are integral to the computing architecture of the PC. The CPU (Central Processing Unit) is an example of a system-level component. System memory and control components and circuits are other examples. System-level hardware in IBM PC, PC AT, PS/2, and compatible computers all have CPUs that belong to the Intel 8086 microprocessor family.

The number of memory locations or addresses that an 8086 (or its counterpart, an 8088) can directly access is 1M (1,048,576 locations). DOS can use up to 640K (655,360) of these memory addresses for executing programs; it reserves the remaining 384K for accessing the operating system. Computers with the more advanced 80286 microprocessor used in the IBM PC AT can address more memory than those with an 8086 or 8088 microprocessor. The 80386

and 80486 microprocessors have even greater addressing ranges. Even on a PC with one of these advanced microprocessors, DOS uses the 1M address range to ensure that it can be used with any member of the 8086 family of microprocessors. Having an 8086 microprocessor is the common hardware characteristic of all DOS-based PCs. In other respects, the system-level hardware of compatible computers can vary considerably. Even with this variation, system-level hardware provides a uniform function.

Peripheral hardware, which connects to the system hardware, provides input and accepts output. Disk drives, keyboards, modems, mice, display terminals, and printers are all types of peripheral hardware. It is the job of peripheral hardware to provide inputs for computing and to display the output of the computing.

Although the system-level hardware "computes," it needs input from the outside world for the computing to be useful. Even a simple keyboard entry is an input for computation. When a computational result is passed from the system hardware, the peripheral hardware outputs the result. If the result is intended to be visual, the peripheral hardware displays it on a printed report or on the screen.

Hardware deals with data at the *binary* or two-state level, and the programs the hardware understands are binary, machine-level instructions. Binary data is ideal for electrical components because the two binary states can be represented by the presence and absence of a voltage. By understanding the interpretation of patterns of on-and-off binary states, computer hardware components can communicate meaningful information electrically. The various display adapters, such as the CGA, EGA, VGA, Hercules, and MDA, each require specific binary instructions to produce output through a display's CRT (Cathode Ray Picture Tube).

Many other peripheral-hardware components and their associated controlling circuitry add to the number of distinct ways that peripheral-controlling software must be configured. At the hardware layer, the electronic specifics of the components can vary from PC to PC. DOS can't be responsible for all of these specifics, so it relies on an intervening layer to handle hardware specifics.

PCs can have both floppy and hard disk drives. Within drive types, a wide variety of configurations can exist. The drives' electronic controllers may follow varying standards and therefore require various control programs for proper operation.

The Basic Input/Output System (BIOS)

The logic that drives and controls the peripheral hardware is a set of low-level, programmed routines called the *Basic Input/Output System* or *BIOS*. The BIOS

deals with hardware specifics. BIOS routines are well-defined, binary-program segments that exploit the peripheral hardware's features and provide a uniform access to hardware functions. Higher-level programs can be built on these BIOS functions. In fact, DOS is built on these BIOS functions. BIOS routines provide an interface between the hardware and programs such as DOS. This interface frees DOS from the need to know exactly how to use the hardware at the binary level.

Cue: *BIOS routines reside in permanent ROM (Read Only Memory).*

BIOS routines reside in permanent *ROM* (*Read Only Memory*), which is located at predetermined addresses within the address range of the CPU. Most ROM is physically located on the PC's main system circuit board, but some ROM can also be located on adapter boards which are plugged into the PC's expansion connectors. DOS supplements the ROM BIOS with extensions, corrections, and additions during the boot process. These extensions are contained in a hidden file on a DOS disk and are loaded during the boot process. The hidden file is named IBMIO.COM, IO.SYS, MIO.SYS, or possibly another name.

Once loaded, the BIOS extensions are located in reserved address areas of the system's *RAM* (*Random Access Memory*). Each version of DOS can include new BIOS extensions to incorporate new hardware or features. The BIOS routines provide DOS with a foundation of elemental operating functions which handle hardware-specific details at the machine level. With machine-level details provided by the BIOS, DOS can concentrate its efforts on providing higher-level operating system services.

The DOS Kernel

The DOS layer of the PC system is a collection of program routines and data tables whose role is to provide higher-level programs, such as applications programs, with standardized and unified PC-operating services. This functional nucleus is called the *DOS kernel*. The kernel resides as a file on the DOS boot disk and is read into reserved areas of RAM during system booting. This file is named IBMDOS.COM, MSDOS.SYS, or possibly another name. Like the BIOS extension file, it is hidden. You do not see the DOS kernel in action; it does its job behind the scenes. The kernel manages the file system, handles character input and output, oversees memory allocation, provides basic networking functions, and performs other operating system tasks. At its lowest level, the DOS kernel calls on the services of the BIOS layer where it uses the BIOS routines as building blocks to supply operating system services to programs. If you want to access the services of the kernel to copy a file or make a directory, you must go through the next outer layer, COMMAND.COM.

The Command Processor (COMMAND.COM)

COMMAND.COM, the command processor program, is a layer of services that rides above the kernel layer of DOS. COMMAND.COM provides a standard set of commands that give users access to file management, device management, configuration, and miscellaneous functions such as maintaining and verifying the time and date. Because COMMAND.COM deals with user commands as a separate module from the kernel, it can be thought of as the kernel's command-interpreting shell. You can recognize this layer of a working computer by the › prompt.

The executable COMMAND.COM file loads during the boot process and accepts commands from the user. COMMAND.COM can execute internal DOS commands, load external DOS commands from program files, and execute the external commands. COMMAND.COM loads and executes DOS batch files and applications programs; it also provides the DOS housekeeping utilities and commands. Chapter 3 discusses COMMAND.COM in more detail and explains other functions of this layer of the system. Part II of this book discusses the internal and external DOS commands. The Command Reference section of this book includes the use and syntax of all the standard DOS commands.

The DOS Shell

Reminder: *Include the DOSSHELL command in your AUTOEXEC.BAT file to have the DOS Shell program loaded during the boot process.*

DOS V4 introduced a keyboard- or mouse-driven graphical user interface that provides many of the functions of the command processor, but with a "point-and-shoot" approach. This windowed graphical user interface is a program named the DOS Shell. The DOS Shell does not replace COMMAND.COM; instead, it provides a substitute for typing commands to COMMAND.COM.

The DOS Shell isn't the same as the term "shell" used to describe COMMAND.COM. The DOS Shell is a program that can be loaded and executed like other programs. The program is not loaded during the boot process, however, unless you include the DOSSHELL command in your AUTOEXEC.BAT file. The DOS Shell is activated by the DOSSHELL command. A user does not need to know how to issue DOS commands at the system prompt to use the DOS Shell. The program provides a visual shell that displays DOS-related actions (commands) in its windows and enables the user to pick a desired action. Chapter 16 is a detailed presentation of the DOS Shell and its practical use.

Applications Programs

The highest level of the working PC system is the layer of applications programs. Applications programs include word processors, spreadsheets, database managers, games, and others. You use many of the programs for your daily com-

puter tasks. Applications programs are written in programming languages such as C, Pascal, BASIC, and Assembler. Programming languages using DOS rely on the services provided by COMMAND.COM and the DOS kernel to provide repetitive file-and-device services, as well as critical-error handling.

Applications programs normally use the DOS kernel interface to access lower layers of the computer. These programs call on DOS to display characters on-screen, access the correct file, and manage the PC's memory resource. DOS, in turn, calls on the BIOS for assistance. Some programs "misbehave" and call BIOS services or access hardware directly. By bypassing the uniform platform provided by DOS, these programs may fail to operate correctly (or at all) on PCs that have some hardware or BIOS variance.

The main reason you use your PC is to run applications programs. These programs seem independent and diverse when they are running. Yet they are only a visible, outside layer of the whole operation. Behind the scenes, the applications programs use DOS and the deeper layers of the PC. Thanks to the layered modularity of the PC and DOS, applications development for PCs is unprecedented. You can select from thousands of programs, but they all rely on DOS.

Summary

In this chapter, you learned some facts about the role of DOS. Following are the key points to remember:

- ❏ The term DOS is an acronym for *D*isk *O*perating *S*ystem. Many computer operating systems have the term DOS in their names.
- ❏ MS-DOS is the name of an operating system for IBM PCs and close compatibles. To most PC users, DOS means MS-DOS.
- ❏ When PC users use the term DOS, they may be referring to only one contextual part of the whole of DOS.
- ❏ DOS works with the Intel 8086 family of microprocessors.
- ❏ A working PC can be viewed as consisting of cooperative layers.
- ❏ The core of a working PC is the system and peripheral hardware.
- ❏ The BIOS layer of a PC is contained in permanent ROM which works with the hardware and provides a foundation of services for DOS.
- ❏ The DOS kernel layer of a working PC is located in RAM and provides the functional nucleus of an operating system.
- ❏ COMMAND.COM executes DOS commands and relies on the kernel for operating system services.

- ❑ The DOS Shell program available with DOS V4 provides a windowed interface that enables the user to execute DOS commands by selecting an item from a menu.
- ❑ Applications programs rely on lower layers of the working PC to provide a uniform platform of control and support.

In the next chapter, you will get a close-up view of the DOS file system. You will examine the hardware, see how DOS does its file "bookkeeping," and gain a fresh perspective on hierarchical directories. Even if you are comfortable with your current insight into the DOS file system, you might find the chapter interesting. If the DOS file system is a gray area for you, you will benefit from covering this material.

2

Understanding the DOS File System

One of the primary roles of a disk operating system is to provide the service of storing and retrieving data to and from disks. The operating system must oversee the many technical details of the disk hardware as well as provide a framework of organization for file storage on disks. This framework includes the way the operating system shows you what it has stored and where it has stored it.

The operating system must also have an internal bookkeeping system to account for unused disk space as well as the contents of used disk space. The operating system should offer a full complement of commands that enable you to manage all aspects of data storage, ranging from disk preparation to performing safety backups. Finally, the disk operating system should provide these services without your needing to know the internal details of the operation.

You shouldn't have to be burdened with technical details in order to save a memo from a session with your word processor. Nor should your word processor have to supervise elemental disk-output tasks to save that memo. A disk operating system assumes responsibility for getting data, in the form of files, to and from a disk file through a predefined organization using predefined commands or methods. The predefined organization is called the *file system*. Disk hardware and file commands are closely linked to this file system. These fundamental disk operations are discussed in this chapter because of this close link. The disk and file commands are discussed in Part II of this book.

Since its introduction, DOS's file system has evolved from simple to more complex (and more versatile). DOS commands that give users access to disk services have evolved as well. Commands have been added to each newer version of DOS, and most current disk-oriented commands take parameters that were not available as a part of the original command.

Many DOS users rely on the older, simpler, and more familiar forms of commands when working with files; some avoid learning new commands. These

users don't take advantage of the full complement of commands that DOS provides for dealing with disks, drives, directories, files, and file-like devices. People are understandably reluctant to master every aspect of DOS file- and disk-related commands when using what they know seems to get the job done.

Without a thorough understanding of the DOS file system, however, some DOS commands can seem cryptic. Once you gain a thorough understanding of the DOS file system, the same commands become more useful.

This chapter is not about how you should use the disk, file, and directory commands. There is nothing in this chapter for you to memorize by rote. Instead, this chapter is a layperson's guide to understanding the DOS file system. First, you will examine files and the disk drives that store them. You will learn about the various disk formats used by DOS. Finally, you will get an inside look at how DOS manages file allocation, directory entry, and partitions. By the end of the chapter, you should have gained a sense of confidence about how DOS commands define and manage the file system and its related support components. While you are using these commands, you should also develop more insight into the hidden DOS kernel layer that was discussed in Chapter 1.

Examining Files and File Systems

Cue:
A file system is an organized collection of files.

A file system is an organized collection of files. As a user, you see the DOS file system at work in the files you organize into directories and subdirectories on your disks. Each file is a named group of data that DOS appears to manipulate as one continuous unit. Behind the scenes, however, DOS employs a structured management strategy whose complexity is hidden from you. The DOS file system reflects this strategy. A file system includes not only the files, but also the internal tables that record the file organization as well as the built-in rules that ensure the consistency of file organization.

Understanding Files

A *file* is a variable-length collection of related information that is referenced by a name. In a concrete sense, you can think of a file cabinet full of file folders, each with a name on the tab. Neither the cabinet nor the the individual pieces of paper in the folders are files. Only the named collection in one folder is considered a file (see fig. 2.1).

Cue:
Disks are the predominant means of file storage for DOS.

In a computer setting, a file can contain data (information), programs (instructions that guide computer tasks), or both. A file can exist on a disk, on magnetic tape, on punch cards, and even in a computer's memory. Files are made of elements of data or programs. The individual elements can be stored as patterns of holes in a paper tape, as patterns of magnetic fields on the surface of a disk, or

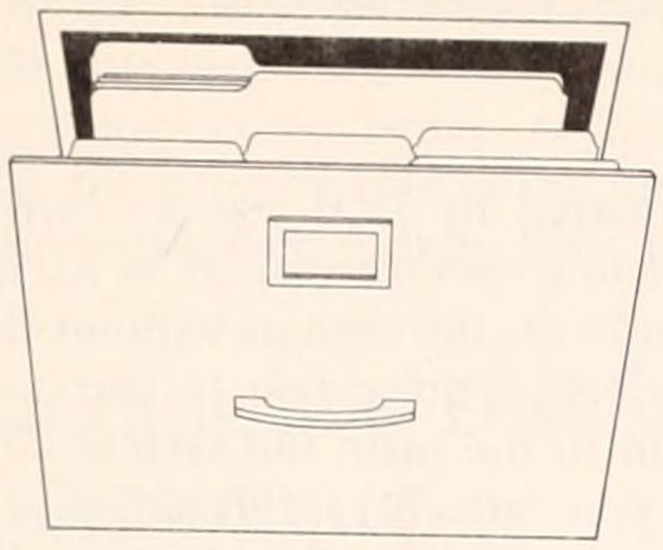


Fig. 2.1

A file represented as a collection of related pieces of paper in one folder.

in many other ways. Physical storage techniques for files vary greatly. In all likelihood, however, when you reach for some stored files, you reach for a disk. Disks are the predominant means of file storage for DOS.

DOS uses file names to identify files. Full file names consist of up to eight characters of a name prefix followed by an optional period character (.) and up to three optional characters of suffix or extension. The full file name is often referred to as the *file name plus extension*.

Cue:
A file name can contain an eight-character prefix (the root) and an optional three-character extension.

The prefix or root of a file name is usually named to describe the contents of the file. Extensions, on the other hand, are traditionally used to describe the type of file. Someone using a word processor, for example, might use as a root file name the name (or a close approximation) of a person who will receive a memo. The memo's extension might then safely be named .MEM. The memo to be sent to Mr. Duncan could be called DUNCAN.MEM. Likewise, the extension .DOC is commonly used for office policy document files. A monthly policy statement might be named JAN.DOC. DOS allows a wide variety of full file names. You are free to develop your own file-naming conventions as long as you stay within the character limits imposed by DOS. Chapter 7 takes a closer look at file names and the rules that govern them.

DOS imposes no structure on the contents of a file. From DOS's point of view, a file is simply a stream of binary information. DOS has no way of knowing if a file is a memo produced by a word processor or a spreadsheet, so it relies on applications programs to apply structure to file contents.

Understanding the File System

DOS is an operating system for personal computers, and personal computers are designed for ordinary people. It is unrealistic to think that the file system used by DOS can have the same complexity as a multiuser, multitasking, real-time minicomputer operating system maintained by a data processing professional. Ordinary users want to use the computer to *do* their work, not to *be* the work. Yet, despite its active base of ordinary users, the DOS file system is ver-

satile. To attain this versatility, the internal structure of the file system is more complex than its surface appearance. In this chapter, you will see below the surface of the DOS file system.

Cue: *Each disk formatted by DOS has its own file system.*

In the strictest sense, each disk formatted by DOS has its own file system. Because DOS commands use a drive-name specifier (such as A:, B:, C:, and so on), you can have the benefit of a multidisk file system without the headache. Some multidisk file systems such as UNIX require you to use a command to "attach" an additional disk's file system to the main file system. You must also "unattach" the additional disk before you "attach" a different disk in the same drive. If you fail to explicitly inform certain operating systems that you have changed a disk in a floppy drive, you can corrupt the operating system's file system. All of this "attaching" is cumbersome and time-consuming.

All disks prepared by DOS are given the essential ingredients for their own independent file system. This disk independence allows DOS to manage a PC with a single disk or many disks. By incorporating named disk drives, DOS can effectively use their file systems as one larger file system.

Cue: *Many DOS commands hide the internal complexity of the file system from the user.*

By hiding the internal complexity of the file system from the user, DOS enables a beginner to use the basic file commands. The COPY command, for example, completely hides the complexity of disk input and output, while the DIR command completely hides the way DOS actually stores files on a disk. In fact, most DOS commands free the user from the need to know what is going on below the user-friendly surface. That's good. You don't need to know how the bendix allows the starter gear to engage the flywheel in order to start your car. Likewise, you don't need to know DOS bookkeeping techniques to copy a disk or a file.

Not every DOS command, hides all the underlying details. The more advanced commands demonstrate this complexity. One example is the CHKDSK command, which reports problems with DOS's internal file-bookkeeping tables. After finding a problem, CHKDSK asks the user to decide whether to alter the tables. Some of the complexities of the file system's details are presented by CHKDSK on-screen, as in the following message:

```
237 lost clusters found in 29 chains.
Convert lost chains to files (Y/N)?
```

Without some knowledge of what CHKDSK is reporting, you can't give CHKDSK an informed response.

Examining Disks and Drives

Cue: *Disks have a virtual monopoly on file storage in PCs.*

The use of files for storing named collections of data is of no consequence without some medium for storing the files themselves. PC designers chose disks as the file-storage medium of choice. Disks, both removable and fixed, have a vir-

tual monopoly on file storage in PCs. Tape and CD ROM (Compact Disk Read Only Memory) finish a distant second and third. Disk drives, DOS's warehouse for file storage, offer convenience, reliability, reusability, and rapid access to files. Disks play the primary role as the file's storage medium, so it helps if you have a good understanding of how disks perform this storage job.

Understanding the Disk's Magnetic Storage Technique

Disk drives are electro-mechanical components that record and play back data using the magnetic surfaces of a disk. Normally, the data is in the form of a file. During the recording or *writing* of a file, an electronic circuit translates the PC's electrical data into a series of magnetic fields. These magnetic fields are mirrored (very weakly) by the oxide coating of the disk in the drive. In effect, the original data is magnetically imprinted on a disk.

During the playback or *reading* of data from the disk's surface, an electronic circuit translates the magnetic fields back into electrical signals, and the data is once again in the electrical form. The disk's original magnetic imprint is not destroyed by the reading operation. Under normal use, only recording over the old imprint, called *overwriting* the disk, will change the imprint. The recorded magnetic imprint on the disk is resistant to weakening and will last for years. Other magnetic fields, however, such as those produced by motors, magnetic paper clip holders, ringing telephones, and televisions, can weaken the magnetic imprint on a disk, and the disk can become unreadable in a short time.

Caution: *Certain magnetic fields, such as those produced by motors, magnetic paper clip holders, ringing telephones, and televisions, can ruin a disk.*

Magnetic recording of electrical signals is not new technology. Recording data information on disks is very similar to recording music information on cassette tapes. There are some differences, however. Because the mechanical parts differ, the media (disks vs. cassettes) that receive recordings differ as well. In addition, the expected quality of operation is different. If the music being magnetically translated from the cassette tape during playback has some momentary loss of fidelity, your ears can tolerate the tape's error. You will still receive the musical information. If the data being magnetically translated from a disk has some momentary loss of integrity, however, your computer will not be able to fill in what's missing or wrong. The result is a *read error*. Read errors can end a work session unpleasantly.

Although designers of cassette tape recorders can be satisfied with "high fidelity" music reproduction from the tape, disk and disk drive designers must strive for perfect data reproduction from computer disks. One bad spot on a disk can result in a `General Failure error reading drive A:` message and the unrecoverable loss of all data on a disk.

Understanding Disk Drives

Disk drives are electro-mechanical computer components. Their mechanical parts and their electrical circuits are complex. Although disk drives are parts of PC systems, the drives are machines in their own right. DOS relies on the driver programs of the BIOS to signal a drive's electronic circuitry to control the actions of the drive's mechanical components.

All disk drives have in common certain components: read/write heads, head positioner mechanisms, and disk spinning motors. Disk drives all record on disks. Some disks are removable; some are built into the drive. Both fixed disks and removable disks are spun on a center spindle within the disk drive.

Many of today's PCs have both fixed disks and removable disks installed in their system units. DOS includes provisions in its BIOS extensions for both types of drives. Even with their common features, fixed and removable disk drives have some important distinctions. Knowing these distinctions will help you understand each drive type's operation in your system.

Hard Disk Drives

Cue:
Drives with built-in disks are called fixed disk drives or hard disk drives.

Non-removable media disk drives have their disks built in. Drives with built-in disks are called *fixed disk drives*. This term is often shortened to *fixed disk* or *hard disk*. Unless the context dictates otherwise, both terms are suitable substitutes for the term fixed disk drive. Fixed disk drives are often called hard disk drives because their disks are made of a hard, rigid metal. In this book, the term *hard disk* is used to describe the built-in, fixed disk drive.

Figure 2.2 shows a cut-away view of a typical hard disk. The circular platters in the figure are the drive's magnetic disks. A head positioner arm holds the read/write heads above and below each surface of each platter. When the drive leaves the factory, the inside components are not open; the drive is sealed closed.

A hard disk drive can contain more than one hard disk or *platter*. Multiple platters are arranged in a stack with space between the individual platters.

Cue:
To move data between the hard disks of two computers, you can use the DOS command XCOPY or the BACKUP and RESTORE command sequence.

Many users refer to hard disks as *Winchester* disks. Winchester is a name derived from an IBM-code name for the sealed disk-drive technology developed by IBM. Hard disks have the advantages of quick operation, high reliability, and large storage capacities. Hard disks have the disadvantage of tying the data stored on the disk to the PC in which the drive has been installed. Because the hard disk's platters cannot be removed, the data is tied to the drive. Hard disks are installed in a PC with mounting hardware and interconnecting power and data cables. Moving an entire hard disk to another computer just to use the hard disk's data is impractical. When you need to move data between the hard

disks of two computers, you can use the XCOPY command, or the BACKUP and RESTORE command sequence. Chapter 6 discusses the use of XCOPY; Chapter 7 discusses BACKUP and RESTORE.

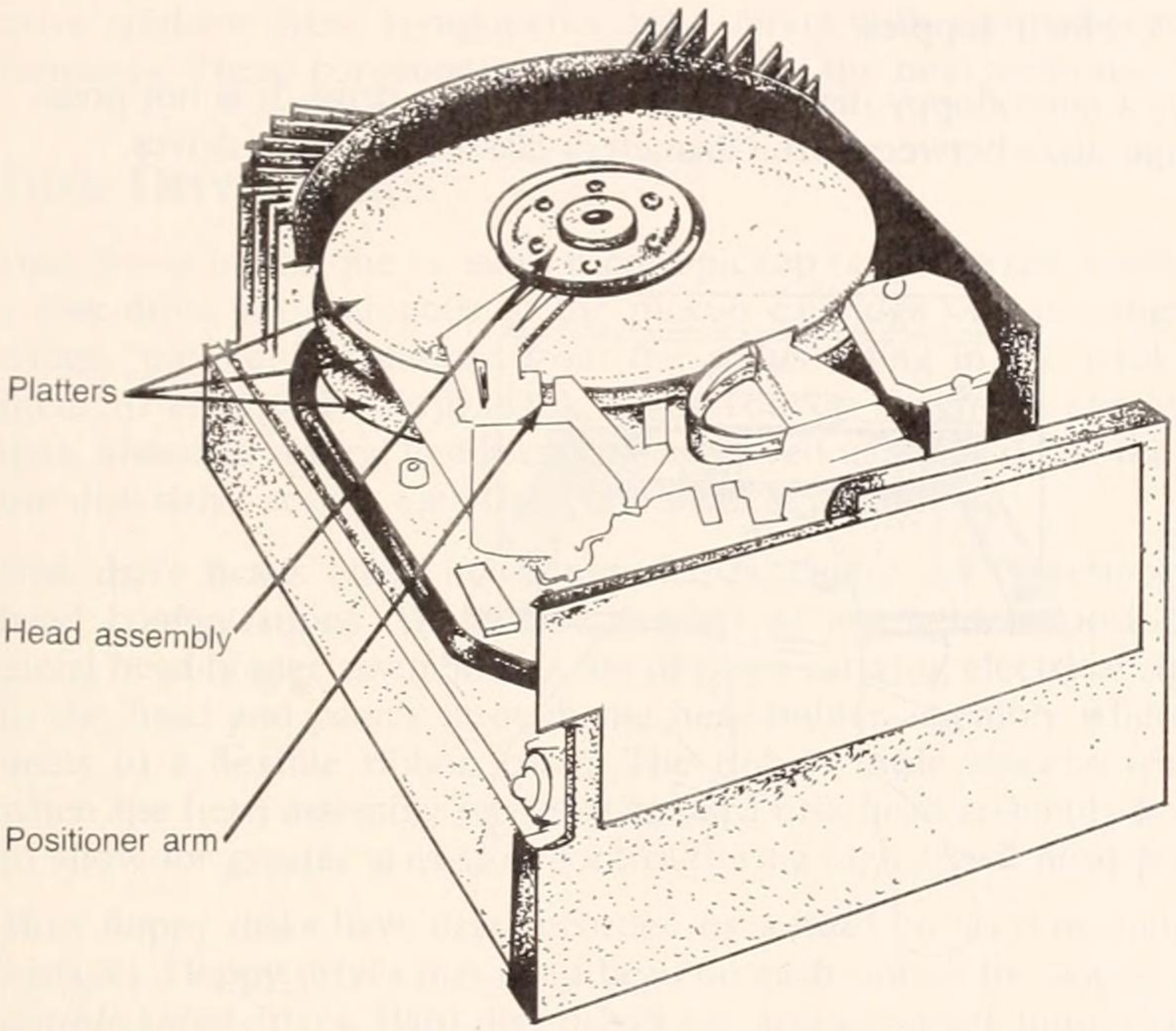


Fig. 2.2

An inside view of the main components of a hard disk.

Floppy Disk Drives

In a PC system, the disadvantages of tying data to the hard disk are compensated by the PC's floppy disk drive. By far the most common class of removable disk is the *floppy* disk. Floppy disks are protected by a permanent jacket which covers the flexible mylar disk and is coated with a magnetic-sensitive film. The first floppy disks were 8 inches in diameter. By today's standards, the early 8-inch floppy disks didn't store much data considering their size. Some 8-inch floppies could only store 320K of data.

Some early microcomputers used these 8-inch floppies as their standard. For three or four years, the pioneering microcomputer makers offered 8-inch floppy drives as the alternative to primitive off-line file storage such as paper tape and cassette recording.

Cue: *A 5 1/4-inch disk is called a minifloppy; a 3 1/2-inch disk is called a microfloppy.*

A smaller version of the 8-inch floppy, the 5 1/4-inch *minifloppy*, quickly became the floppy of choice for PC designers because of its size. The 3 1/2-inch *microfloppy* is yet another departure from its larger, older cousins because it incorporates a rigid plastic case (its jacket) as a protective cover. The 3 1/2-inch mylar circular medium inside a microfloppy disk is flexible like the media in 8-inch and 5 1/4-inch floppies.

Figure 2.3 shows a microfloppy disk being inserted into a drive. It is not possible to interchange disks between size (diameter) classes of floppy drives.

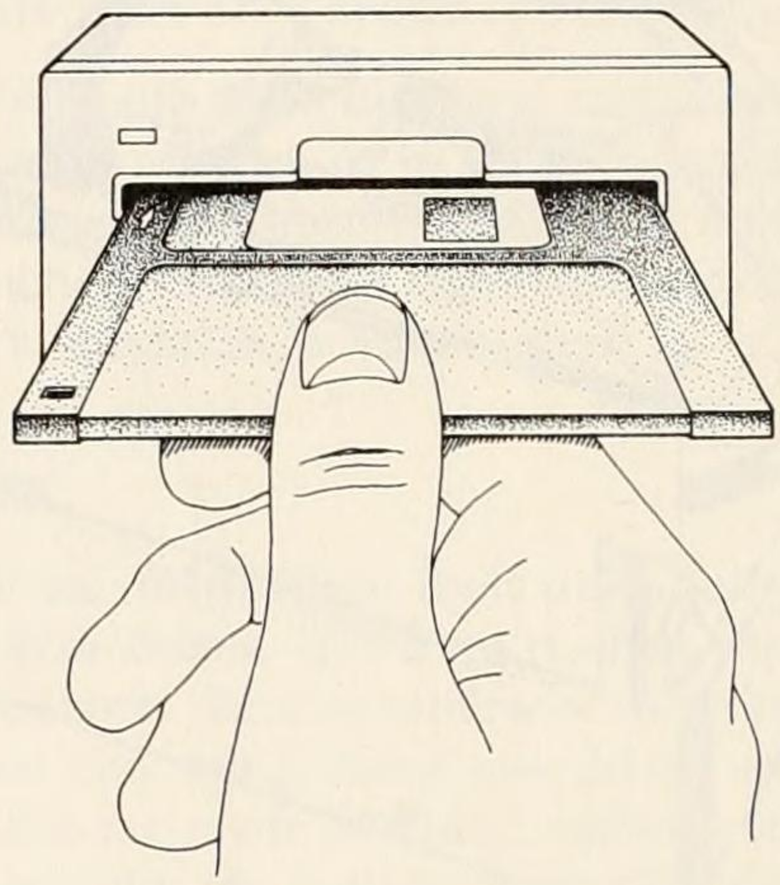


Fig. 2.3

Inserting a microfloppy (3 1/2-inch) disk into a drive.

The smaller two sizes of floppy disks are officially called *diskettes*; the 8-inch version is called a *disk*. The term *floppy disk* (or just *floppy*) refers to any lower-capacity disk in a jacket that you remove from your PC's drive. If identifying the type of floppy is important, many users use the size qualifier as in "3 1/2-inch floppy" to make a distinction. *Minifloppy* always refers to 5 1/4-inch floppy disks; *microfloppy* always refers to 3 1/2-inch floppy disks. You'll likely read or hear the term *disk* used to mean any floppy or hard disk when the disk type is clear in context. In other contexts, the term *diskette* is used to refer to both 3 1/2-inch and 5 1/4-inch disks.

Understanding the Dynamics of the Disk Drive

The dynamics of the disk drive are a function of the drive's electronics. The drive contains many components that interact with each other to provide performance. These components are covered in the next sections.

Disk Drive Heads

Disk drives utilize one or more record/pickup or read/write *heads*. The heads of a disk drive are analogous to the pickup cartridge of a phonograph. The cartridge "picks-up" vibrations from the stylus riding in the track and converts them to electrical energy. Disk heads convert magnetic energy to electrical data. Although several heads can be employed in a disk drive, the electronics of the disk drive accept data from one head at a time.

Cue:
Disk heads convert magnetic energy to electrical data.

Disk drive heads come in various shapes. Figure 2.4 illustrates two common head configurations. The heads themselves are held in position by flexible metal head-holder assemblies. A set of wires carrying electrical signals connects to the head and passes through the head-holder assembly where the set connects to a flexible ribbon cable. The ribbon cable absorbs wire movements when the head assembly moves. The hard disk head assembly is lower in mass to allow for greater start-stop control during high speed head positioning.

Most floppy disks have data recorded by a head on each of their two sides or surfaces. Floppy drives that use a head on each side of the floppy disk are called *double-sided* drives. Hard disk drives can accommodate more than one double-sided platter because the drives incorporate heads on both sides of each platter. Using both sides of a disk or platter doubles its capacity. Incorporating multiple platters in a hard disk further multiplies its capacity by the number of heads employed.

Disk Tracks

Regardless of the type of disk drive, all disks spin on a center axis like a record spinning on a phonograph. A floppy disk spins at 360 revolutions per minute. The rotational speed of a hard disk is 10 times higher (approximately 3,600 RPM). The heads, which are positioned above the spinning surface of the disk, are held and moved in distinct steps by an actuator arm and head positioner. The heads of a floppy disk drive touch the medium's surface; the heads of a hard disk ride above the surface of the media on a cushion of air. At each step position, the alignment of the head and the spinning disk produces a circular track.

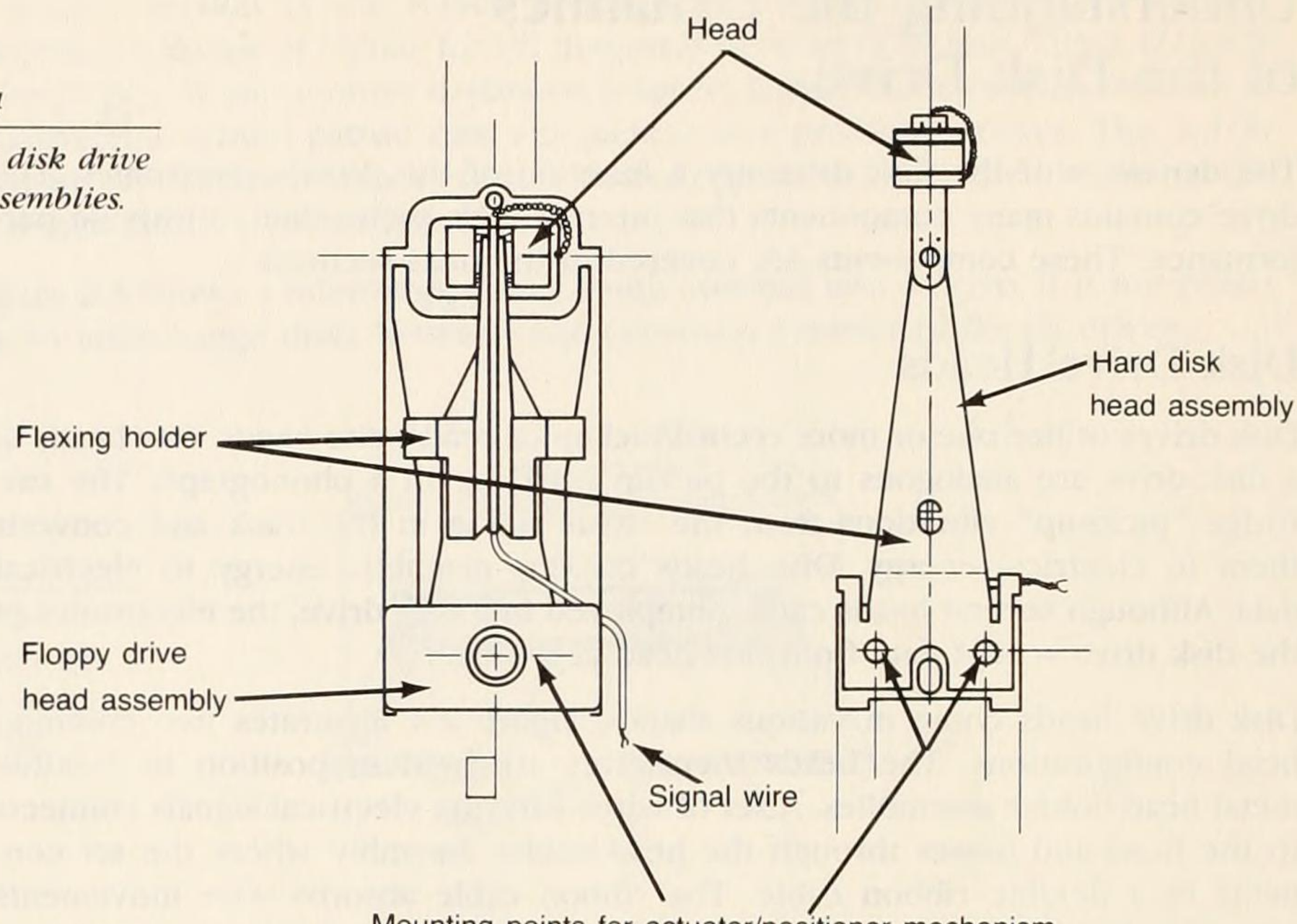


Fig. 2.4

Typical disk drive head assemblies.

Cue:
All the tracks on a disk's surface are perfect concentric circles.

The track is not a physical track like the groove in a record, but rather a ring of magnetically recorded data. Unlike the phonograph, which plays a record's contents by following a single, spiraling track, the disk drive steps from track-to-track to retrieve data. Figure 2.5 gives some idea of the position of the tracks. Notice that the tracks are not depicted as a spiral. All the tracks on disk's surface are perfect concentric circles. The number of concentric tracks available on one disk's surface is determined by the head positioner's mechanical movements. The smaller the stepping distance, the tighter the track pattern, and consequently, the greater the number of tracks.

Cylinders

A disk drive's multiple heads are affixed to a single positioner mechanism. When one head moves one track on its side of a platter (or floppy), the other heads all move one track on their respective sides of their respective platters. Picture a top head riding over track 10 of side one of a platter, while the bottom head is riding under track 10 of side two. If the disk has more than one platter, all the heads of that disk are positioned on track 10 of their associated side and platter. This conceptual alignment of heads on the same track position

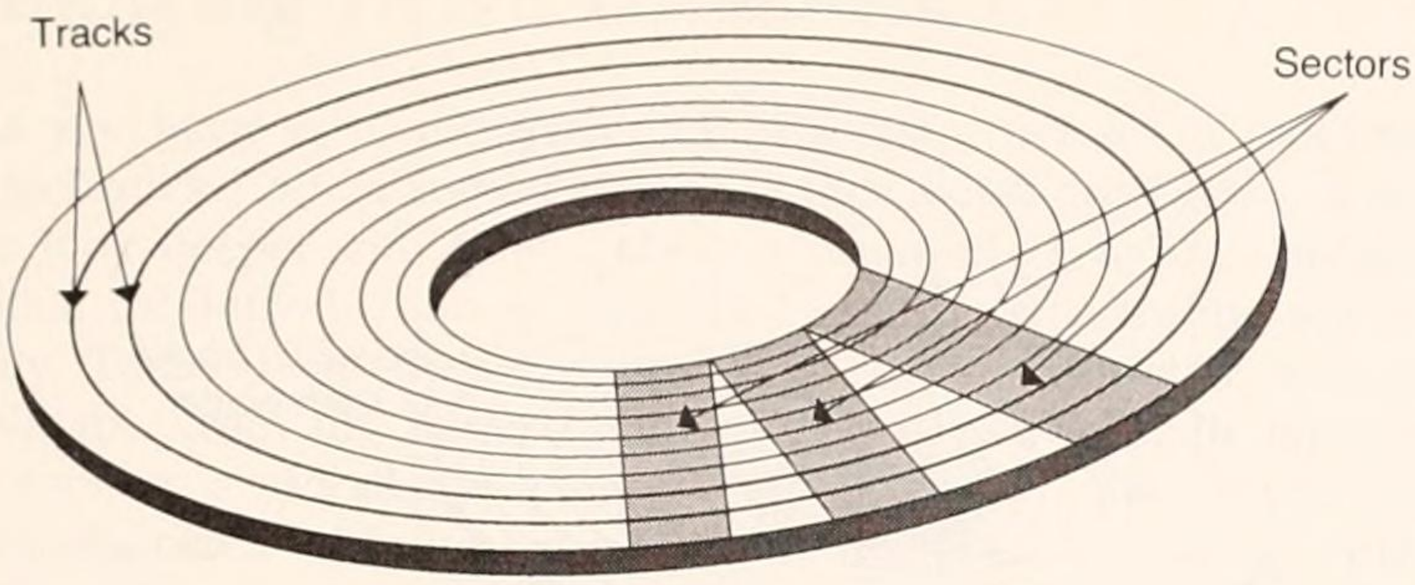


Fig. 2.5

Concentric tracks on a disk's surface.

on different sides (and platters) of the same disk is called a *cylinder* (see fig. 2.6). The term cylinder is derived from the imaginary shape of the stacked circular tracks at any one stopping point of the positioner mechanism.

Because only one head can be active at one time, the drive must activate all of its heads in sequence to write (or read) all tracks at the cylinder position. Head activation starts at the top head. To fill a cylinder, a 4-head drive writes a track with head 1, then head 2, then head 3, and finally head 4. The head positioner moves one cylinder, and the sequence repeats. Processing all tracks of a cylinder before moving to the next cylinder is efficient because all heads are already at new track positions. You will see how DOS uses this efficiency in a moment.

Seek Time

The time it takes a drive to move the heads from one cylinder to another cylinder is called its *seek time*. Hard disk drives can move the heads from one cylinder to another quickly and therefore have low seek times. On the average, a hard disk can seek to a cylinder within 65 milliseconds. Some higher performance hard disks can perform the same seek in 10 milliseconds. The seek times of floppy disk drives can be over 10 times slower than the seek time of a slow hard disk.

Seek time is an important measure of disk drive performance. Often, the heads must seek several nonadjacent cylinders to collect the parts of a file. You may have noticed that many applications programs seem sluggish when they use a floppy for file operations, but when they use a hard disk, the same programs seem quite speedy. Seek time, rotational speed, and the nature of the drive's interfacing electronics all affect a drive's relative access time performance.

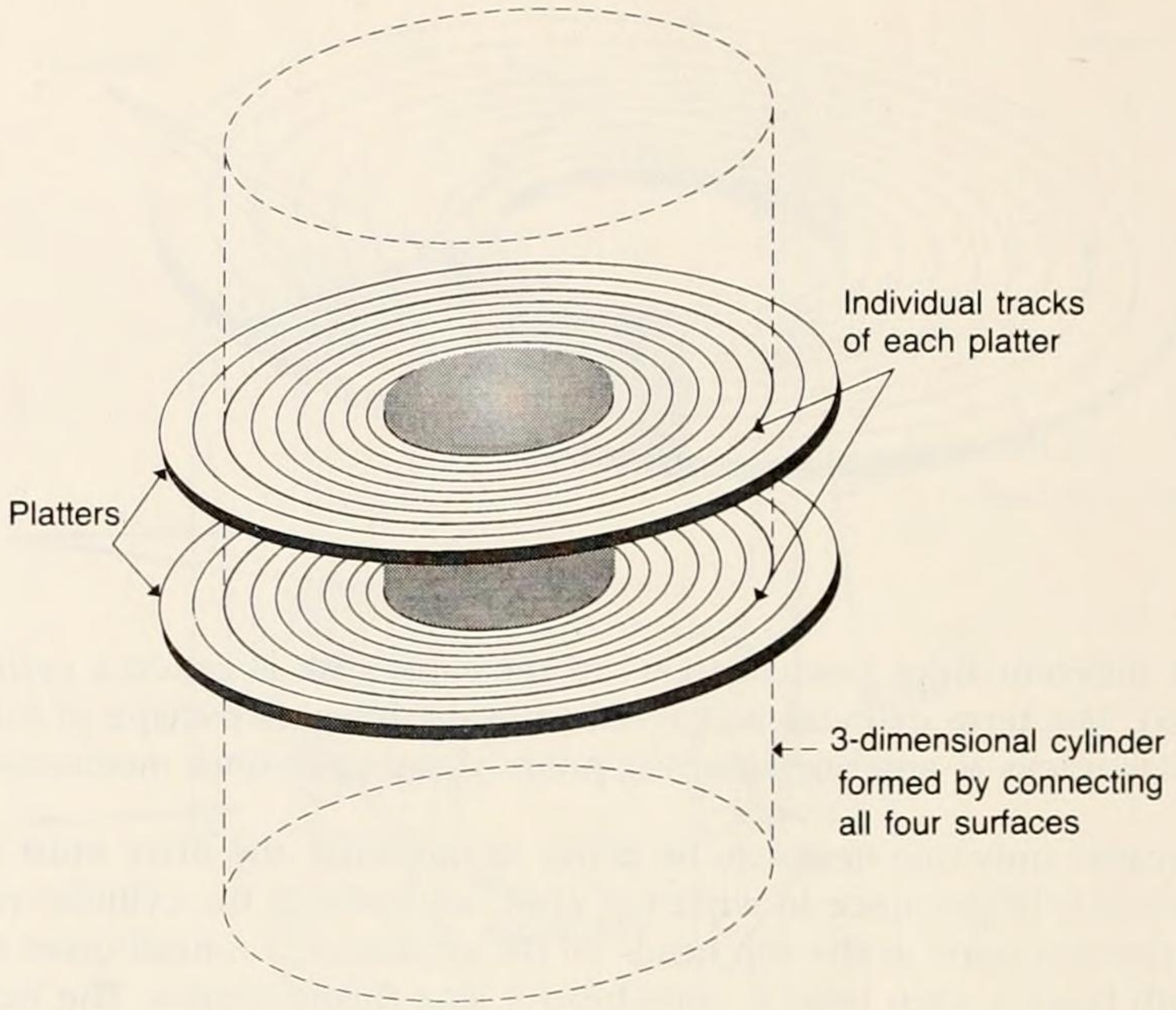


Fig. 2.6

A cylinder passing through two disk platters.

Disk Drive Alignment

Floppy disks do not come with their tracks (or cylinders) identified when they leave the manufacturer. It is the disk drive's job to put the data in tracks. Furthermore, the disk drive has no grooves or markings to use as a guide for where a track should be. To ensure interchangability of floppies between different drives (of the same type), the drives must be aligned to a standard.

Technicians use alignment disks, which contain carefully positioned tracks, to achieve this standardization. With the alignment disk in the drive, the technician adjusts (mechanically or electrically) the positioning mechanism of the drive until the heads are one-to-one with the special tracks of the alignment disk. Drives are designed so that once aligned, they resist "drifting" out of alignment. A drive that has become out of alignment will lay down tracks of data on a new disk and read them back with no error. The alignment problem usually becomes apparent when a different drive attempts to read the disk from an improperly aligned drive. Heads will not align to track positions, and read errors may result.

Applying Drive Dynamics to Data Storage

Cue:
Disks of all sizes and capacities are organized by DOS with the same file system approach.

As you have seen, the dynamics of a disk drive are a function of the drive's mechanics. Understanding how DOS uses a disk drive to store data requires you to look deeper into how disks lend themselves to standardized data formats. Since the introduction of the IBM PC, disk drive technology has evolved rapidly. Today, even with the same disk dynamics, higher storage capacities are available from the same size of medium. DOS, from its layers of the working computer, sees a disk as a logical device. In DOS, disks of all sizes and capacities are organized with the same file-system approach. In your work with the computer, you see a disk as a file system because DOS connects you or your program to the working computer as a level above the details of disk technology.

Before examining how DOS sees a disk, you need some background on how disks work with data. The next few sections explore disk drives as the primary medium of data storage. This section builds on the previous section by considering how drives record data and why drives use standard ways to record the data. When you are ready to examine file systems from DOS's point of view, you will understand how DOS bases its file system on disks.

Track Density

Getting more storage capacity from the same disk is desirable. One way to get more capacity from the same disk size is to put the tracks closer together on the disk so that the same disk contains more tracks or cylinders. With more tracks, the disk can hold more data. Why not, then, dramatically increase the number of tracks on the disk's surface? The number of tracks can be increased, but there are limits.

To understand how the drive head records information as a track on the disk's surface, picture the drive's recording head as a tiny spray-painting nozzle whose paint is a spray of magnetic energy. The disk's surface with its magnetic particles is like a wooden wheel with its grain and texture. As the wheel spins, the spray nozzle directs its paint in a narrow stream onto the surface of the wheel.

After the wheel has turned one complete revolution and the spray nozzle turns off, a perfect circle is painted on the wheel's surface. It is in this painting manner that the drive's head(s) magnetically alter the disk's surface. The nozzle is stepped-in just a small amount toward the wheel's center, and the spraying process is repeated. Now there are two tracks of paint on the wheel's surface. By continuing the step-in and paint process, the wheel will eventually have its surface completely painted with concentric circles. Likewise, a disk will have its available surface divided into distinct data tracks.

If you want to double the number of rings on the wheel's surface, you can repeat the entire operation of painting and stepping to the center of the wheel.

This time, you can step the paint nozzle toward the center by one half the distance per step that you used the first time. The process includes twice as many steps and results in twice as many painted rings. Of course, the same doubling of tracks results from the same reduction of step size between tracks of data on a disk. The same disk has its capacity effectively doubled. It is a double-density disk.

There is a limit to the number of distinct rings this capacity-increasing trick can allow. The rings will eventually get so close together that you can't tell where one ring ends and the next ring begins. The rings blur together and lose their individual identity. On the surface of a disk, the magnetic fields of the tracks begin to interfere with adjacent tracks. The effect is similar to the distortion that results from tuning a radio between two close signals. To reduce the potential of blurring on the wooden wheel, you can change to a paint nozzle that produces a finer paint stream, or you can smooth the wheel's wooden surface so that the grain is finer and less prone to allowing the paint to bleed. This process results in more, but fainter rings.

In the disk drive, you can employ a head with a smaller magnetic gap designed to produce a narrower magnetic field, or you can use disks with very fine-grained magnetic particles that resist magnetic bleeding. By employing more precise technology in this way, you can get more tracks on the disk; moreover, each track can retain its individual identity. The more precise head and disk surface produces a weaker magnetic imprint which requires more electronic and mechanical alignment precision to read the weaker data tracks. Even with the additional precision, there is a point at which the tracks begin to blur together as a drive makes smaller steps between tracks.

Data Density

Data, however, is not recorded in tracks as a constant stream of magnetic signals like a constant spray of paint. Instead, data is written as a series of magnetic energy pulses which represent the data. During data reading, the magnetic fields created by the pulses are deciphered by the drive's electronics and converted back into data.

When the drive is writing (recording) data, the pulses are like magnetic bursts created by the head. These bursts are registered in a track as it spins by the head. The information in a file is encoded as a series of off-and-on patterns of these magnetic pulses. If the magnetic pulses were paint, they would resemble dots. In theory, if you increase the number of pulses in a track, you increase the capacity of the disk. In other words, if you want more dots, put them closer together. By packing two pulses in the track space that holds one, you double the capacity of a disk.

You can pack more pulses in a track by reducing the duration of each pulse. This reduction in pulse duration is analogous to spraying shorter bursts of paint.

Shorter bursts produce smaller paint dots which can be painted closer together. The same is true for disk data. If you reduce the space it takes to hold one bit of information, you can hold more bits of information in any given space.

There is a limit, however, to how tightly you can pack magnetic fields in a track before an individual field begins to blur into its neighboring fields. If the magnetic information is too tightly packed, the drive's electronics can't properly read it. Precision heads and special magnetic disk coatings help increase data density in each track, but even with these precision components, there are limits to disk technology.

Disk Sectors

When a disk is blank, as it is when it comes from the factory, it contains no tracks and therefore no information. DOS has to prepare the disk to accept data. You know this preparation exercise as formatting. Formatting, as the name implies, puts a uniform pattern of format information in all tracks as it steps through each cylinder of the disk. This format information within each track allows DOS to slice each track into smaller, more manageable, fixed-size component parts. These component parts are called *sectors*. Figure 2.7 shows the sectors of a floppy disk represented as slices of the disk's surface. The figure shows the boundaries for every other sector. The concentric arcs between the indicated sectors are the disk's tracks. Notice that each track has the same number of sector boundaries (and therefore sectors).

Cue:
The DOS format process puts a uniform pattern of format information in all tracks as it steps through each cylinder of the disk.

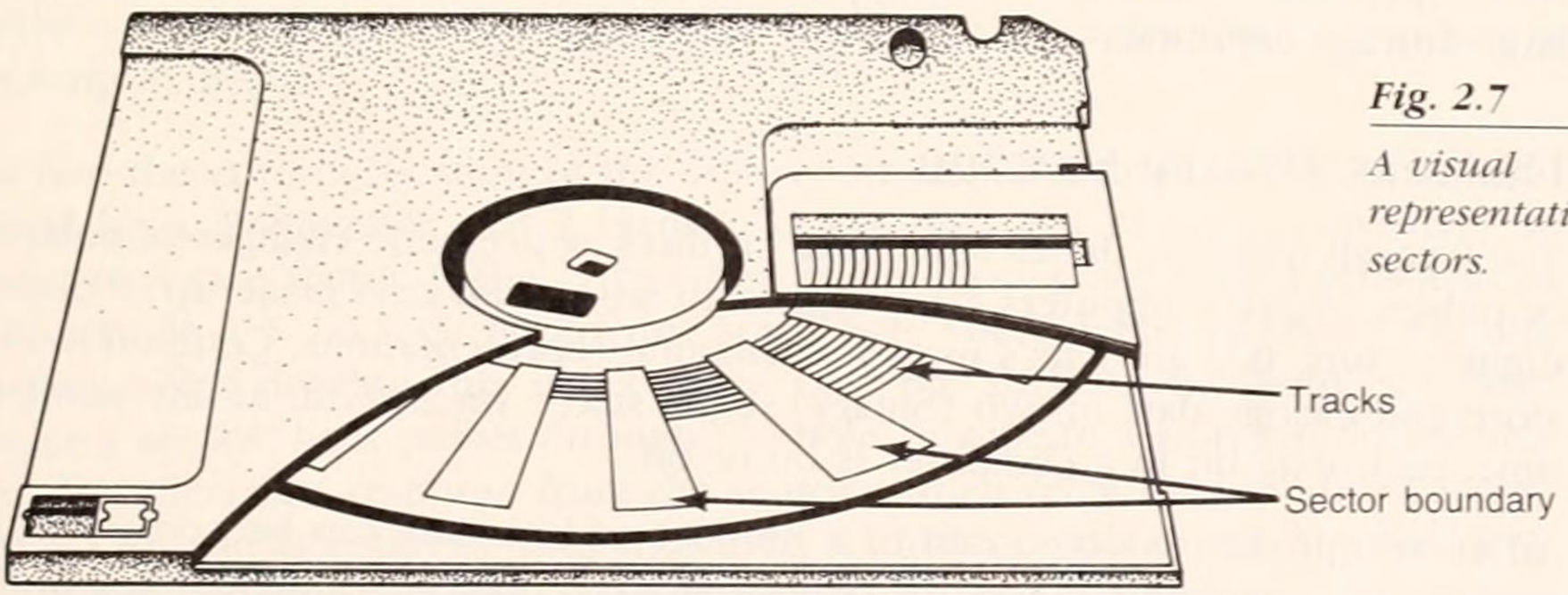


Fig. 2.7

A visual representation of sectors.

DOS prefers to read and write data one sector at a time. Some of its internal file-system bookkeeping is done by sectors. To DOS, a sector is the disk's most manageable block of data. DOS uses 512-byte sectors with disks, but DOS has provisions to use a different sector size. DOS has been using 512-byte sectors since its beginning, so a change in sector size is unlikely (but not impossible) in the future.

Cue:
To DOS, a sector is the disk's most manageable block of data.

The number of sectors formatted into each track is tied to the data density that the drive uses when reading or writing data. The more dense the recording in a track, the greater the number of sectors that can be formatted. The more tracks, the greater the number of sectors per disk.

Designers select the number of tracks and the number of sectors per track with reliability in mind. Floppy disk drives are designed with more margin for error than hard disks. It's easy to see why some margin for error is desirable for a drive that must ensure the mechanical alignment of each disk a user shoves in the door. Floppy disk drives must read disks that may have been stored in a place where the disk's magnetic information has been weakened. Even with the protective jacket, the magnetic-coated surfaces of many floppy disks become contaminated with smoke, dust, or finger prints. The drive must be able to tolerate some contamination on a disk and still perform its function without numerous errors. Of course, no disk drive can avoid errors if the disks it uses are abused. Drive heads can't read through liquid spills or ballpoint pen dents.

Hard disk drives have higher capacities than their floppy cousins. The higher capacity is due in large part to the precision of the drive's critical components in conjunction with the special oxides used to magnetically coat the platters. In addition, the working parts of the drive are sealed at the factory in a way that protects the platters, positioners, and heads from particles and contamination. With outside influence over these critical components sealed out, the hard disk drive can offer more tracks and sectors in the same physical space. When you consider that most hard disks have more than one platter, each capable of two-sided operation with more tracks, you can begin to see how hard disks get their large storage capacities.

Disks as Binary Storage

Cue:
Binary digits or bits are the most fundamental representations of data in a computer.

You'll recall that disk drives store data on disks as magnetic changes of polarity or pulses. Digital computers are designed to work with data made up of binary digits or *bits*. Bits are data's most fundamental representations. Computers can store and change data in two (binary) stable states: on and off. At any point in time, each data bit in a computer is on or off.

All computer data is composed of a number of bits. Bits can be conveniently converted to magnetic pulses for reliable storage as magnetic fields on a disk's surface. Magnetic fields have two poles: north and south. When a disk drive head cuts through the polarized magnetic fields of data on the disk, a positive or negative voltage is produced in the head's coil. The drive electronics treat this induced voltage as either a binary 0 or 1. Although this explanation is a simplified view of disk magnetics, you can consider that the data in a magnetic field directly relates to the binary data in a computer. The coils in the head change the state of or *transduce* the magnetic energy to binary data through the drive's electronics.

Bytes and Disk Capacity

Cue: *Computers generally store data in groups of eight bits. This eight-bit group of data is called a byte.*

Computers generally store data in groups of eight bits. This eight-bit group of data is called a *byte*. By design, digital computers are most efficient when working with numbers as some power of 2. Numbers that are powers of two can be represented directly in a binary annotation.

Computer programmers and designers apply this power-of-2 convention to the expression of a quantity of bytes. A kilobyte, for example, is 2^{10} or 1,024 bytes. 2 kilobytes = 2,048 bytes or 2K. A megabyte, or 1M, is 1,024K. It's not important for you to know about numbers in the power of 2 except to note that capacity in kilobytes or megabytes has this 1,024 rather than 1,000 multiplier. You can, for scaling purposes, think of 1K as approximately 1,000 bytes.

The capacity of disk drives is stated in bytes as either kilobytes or megabytes. The storage capacity of a hard disk is in the order of millions of bytes; megabyte is the usual capacity descriptor. The capacity of floppies is in the order of hundreds of thousands to just over a million bytes. Both kilobyte (K) and megabyte (M) can be used as a multiplier to describe the capacity of a floppy.

Reminder: *During formatting, each cylinder is segmented into 512-byte sectors.*

The sectors in an individual disk store a fixed number of bytes of data. This fixed number can vary from operating system to operating system or even from disk to disk. In DOS, the size of the sectors on all disks is 512 bytes. 512 bytes just happens to be 1/2K or 2^9. DOS provides for other sector sizes, but 512-byte sectors are the rule. During formatting, each cylinder is segmented into 512-byte sectors, starting with the first sector of the first track (the top head) and working to the sector of the last track (the bottom head). Remember that a drive is more efficient if it completes all track work at each cylinder stop before moving its heads.

To find the correct location of the first sector, the disk drive needs some additional help. Remember that a blank disk has no track or sector divisions. In order to locate sector 1, the drive relies on an index pulse from a sensor in its circuitry. In a 5 1/4-inch floppy, the small hole you sometimes see in the disk surface helps provide this index pulse (see fig. 2.8). As the hole is rotated under a sensor, light passes through the hole to actuate a response in the sensor. The electrical response from the sensor is then relayed to the drive's electronics for use as a start of track indicator. With this reliable indicator, the drive can begin to record sector 1's information.

3 1/2-inch and hard disk drives rely on index pulse indicators; the method of sensing, however, isn't exactly the same. By detecting sector one of track one, the disk drive can extract some specially coded data that helps DOS determine how to manage the disk. The one location on every disk that DOS is guaranteed to find is this first track and sector. It will always be at cylinder one, head one, and immediately after the index pulse. Once the special data is known, DOS can determine the location of any other sector on the disk.

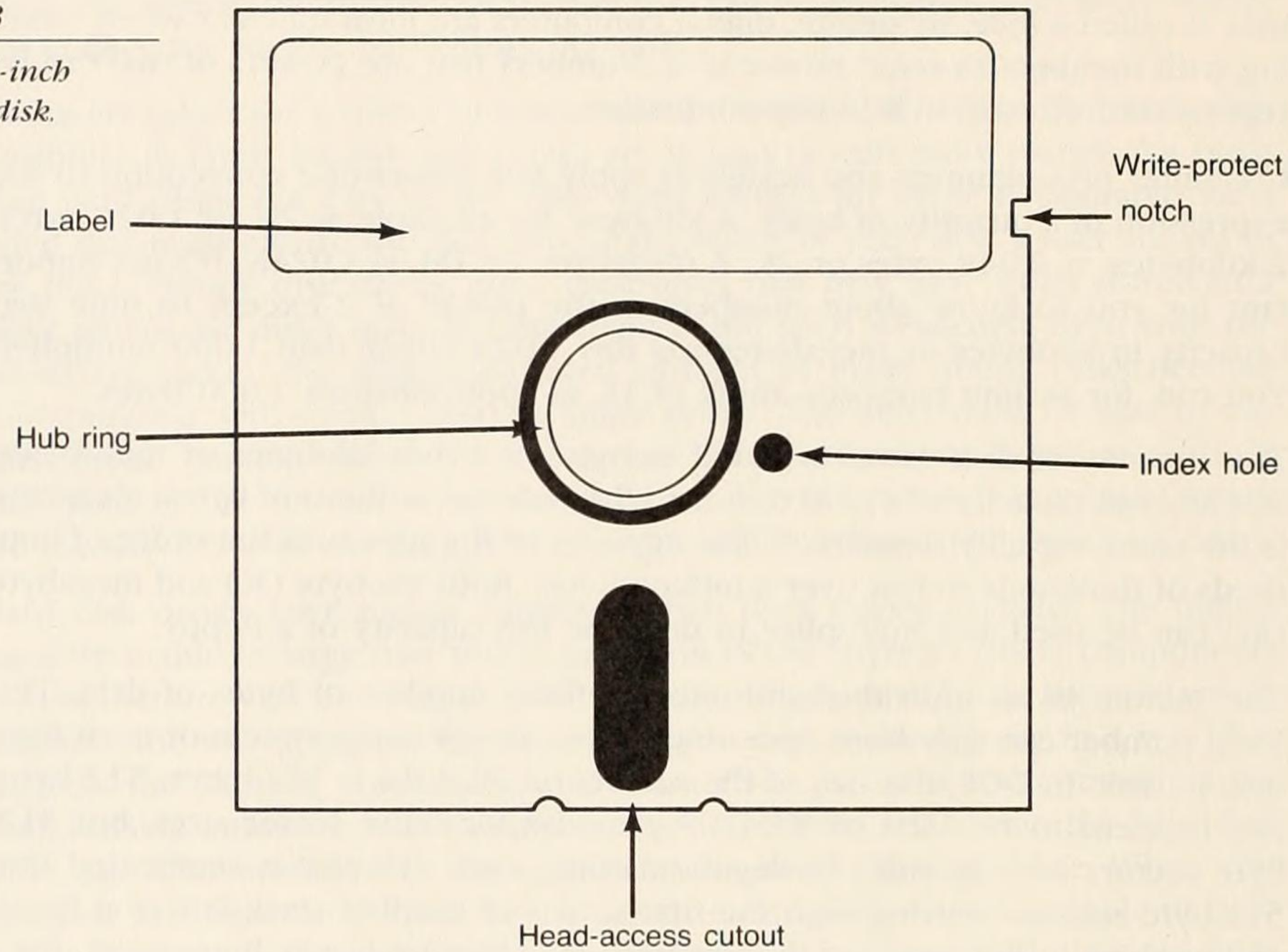


Fig. 2.8

A 5 1/4-inch floppy disk.

Understanding Disk Formats

Disk drives have a universal way to divide a disk's available physical space in a logical manner. The number of platters, the number of sides, the number of tracks, the number of bytes per sector, and the number of sectors per track are the specific details that allow this logical division of a disk's physical space. The specification for a disk's use of its physical space is called its *format*. PCs employ a variety of disk drive sizes and formats. PCs equipped with both 5 1/4-inch and 3 1/2-inch floppies are available. Most PC users and software manuals differentiate formats by using the byte-capacity figure for the desired format. Each new version of DOS has maintained support for disk formats supported by its predecessors. This support ensures that disks made with older drive formats will be usable with current versions of DOS.

Floppy Drive Formats

The first DOS-supported disk drives allowed for double the number of tracks on a 5 1/4-inch floppy disk than the standard 5 1/4-inch disk formats of the time.

These DOS formats were called double-density formats. The original PC disk size and format was 5 1/4-inch, single-sided, 40 tracks with 8 sectors per track, and 512 bytes per sector. These disks are called single-sided, double-density disks or SSDD. The capacity of this 8-sector single-sided format is 160K.

Cue: *During formatting, data is recorded on both sides of a double-sided disk.*

The early format was extended by making the disk format double-sided in DOS V1.1. All floppies are double-sided in the sense that they have two sides. The term *double-sided* in the formatting sense means that data is recorded on both sides of the disk. Only drives equipped with a second head can accommodate double-sided recording. By using the other side of the disk, disk capacity is doubled. To differentiate these two-sided disks from disks that used only one side for storage, disk makers called them double-sided, double-density disks or DSDD. By using both sides for storing data, the format capacity was 320K. Today it is rare to find a PC with a single-sided disk drive. The majority of 5 1/4-inch floppy drives are equipped with two heads.

As disk drives became more sophisticated in design and magnetic materials made great improvements, the number of sectors per track was increased from 8 to 9 in DOS V2.0 with no reliability problems. Both DSDD and SSDD formats were given the extra sector per track. The extra sector per track allowed this format to store more data than the previous DSDD and SSDD formats, and the new format quickly became popular with users. To differentiate between the DSDD and SSDD, 8-sector formats, and these DSDD and SSDD, 9-sector formats, you can think of these formats as DSDD-9 and SSDD-9. Of course, referring to the DSDD and SSDD, 8-sector formats as DSDD-8 and SSDD-8 provides additional clarification. The single-sided, 9-sector version has a capacity of 180K; the double-sided version has a capacity of 360K.

The evolution of DOS to V3.0 provided for drives with quadruple the number of tracks of those early standard disks. These new 80-track quad-density formats were applied to 5 1/4-inch drives as well as the newer 3 1/2-inch drives. DOS provided one quad-density format of 9 sectors per track, used primarily on 3 1/2-inch drives. This quad-density, 9-sector format is called QD-9, and sometimes just quad. The QD-9 disk capacity is 720K. A second quad-density, high-capacity quad format of 15 sectors per track was incorporated by DOS designers at the same time. This high-capacity format is used primarily on 5 1/4-inch drives. The quad-density, 15-sector format is called HC for High Capacity even though it is just as proper to refer to it as QD-15. The QD-15 format, with a capacity of 1.2M, was popularized by the IBM PC/AT.

DOS V3.3 added a high-capacity format for 3 1/2-inch drives that supports the 80-track quad density but provides 18 sectors per track. The high capacity format offers 1.44M of storage space from a microfloppy. This QD-18 format is sometimes called 3 1/2-inch HC.

Table 2.1 summarizes the common floppy disk formats. Some versions of DOS enable you to use the FORMAT command to place a format on a disk other than the normal format. Chapter 5 explains the details of the FORMAT command.

Table 2.1
DOS Floppy Disk Formats

Format	*Tracks*	*Sectors Per Track*	*Total Sectors*	*Usable Capacity*
SSDD	40	8	320	160K
DSDD	40	8	640	320K
SSDD-9	40	9	360	180K
DSDD-9	40	9	720	360K
QD-9	80	9	1420	720K
QD-15	80	15	2400	1.2M
QD-18	80	18	2880	1.44M

Raw Capacity and Usable Capacity

The process of formatting a blank disk places some data on the disk that is not part of the total capacity of the disk. A 1.44M disk, for example, holds more than 1.44M bytes of information. This extra space is not available to you, though. It is the space used by the sector-identification and error-checking information. If you buy disks for a 1.44M drive, the disk's identification label may say that the disk has a 2M capacity. Disks for a 720K drive may indicate a 1M capacity.

This apparent discrepancy is explained by looking at the difference between total or raw capacity and usable or formatted capacity. The larger of the two numbers for the same disk is considered the *raw* capacity of the disk. Raw capacity includes the capacity that the formatting information will use at format time. The lesser of the two numbers for the same disk is the *usable* capacity of the disk. This number of bytes is available for storing files after the formatting information has been put on the disk.

Fortunately, most hard disk manufacturers state the capacity of their drives as the formatted capacity. Hard disks also lose some overhead space. If you have any doubt as to the meaning of a hard disk's stated capacity, ask the dealer if the capacity is determined before or after formatting. In this book, disk capacity refers to usable capacity after formatting-overhead information has been applied to the disk.

Hard Disk Drive Formats

Formats for hard disks nearly always employ 17,512-byte sectors per track. You can understand the concept of hard disk capacity by remembering the concept of cylinders. Hard disks, as you recall, have two or more heads. Remember that

a cylinder is the alignment of all of the heads on the same track on both sides of each platter. If a disk has 306 tracks on one side of one platter, the disk has 306 cylinders. The total number of tracks on the disk is the number of cylinders times the number of heads. The disk's capacity in bytes is the number of tracks times the number of sectors per track, times the number of bytes per sector. To obtain the capacity in kilobytes, divide the result by 1,024. To obtain the capacity in megabytes, divide the kilobyte total by 1,024. For approximations of capacity, you can divide by a rounded 1,000.

Cue: *The DOS FORMAT command can perform a high-level format on a hard disk.*

DOS does not actually provide the low-level format data for a hard disk as it does a floppy disk. Hard disks are normally prepared at the factory with a low-level format. End users seldom (if ever) need to initiate a low-level format on a hard disk. DOS uses the low-level format to perform its high-level format. When discussing hard disk formatting, the term *format* refers to the high-level format initiated by the DOS FORMAT command. During the formatting of a hard disk, DOS initializes its bookkeeping tables and then writes "dummy" data into the disk's tracks. From your point of view, formatting a hard disk is the same basic operation as formatting a floppy. DOS keeps the details of the low-level format hidden and out of your way.

Table 2.2 shows some typical hard disk formats. An increasing number of hard disk sizes are being used in PCs, and each size or type uses the same basic principle of format.

Table 2.2
Hard Disk Formats

Typical Disk	*Sectors Per Track*	*Heads*	*Cylinders*	*Capacity*
IBM PC/XT	17	4	306	10M
IBM AT	17	4	615	20M
IBM AT (Late Model)	17	5	733	30M

Examining How DOS Manages Disks and Files

You can think of blank disks as empty warehouses in which DOS intends to store files. Floppy disks are relatively small file warehouses; hard disks are large warehouses. If you enter a completely empty warehouse, you see nothing but a shiny floor. No material is stored in the empty warehouse, and the floor space has not yet been organized with aisles, shelves, or storage bins.

This empty space is an example of how DOS sees a blank disk. The disk has no tracks, cylinders, or sectors. After DOS formats the blank disk, the heads, cylinders, and sectors act as aisles, shelves, and bins for data organization and storage. To DOS, receiving a new file is like receiving a shipment of particular product. The quantity of the file's data bytes is like the shipment's quantity. When DOS stores files, DOS places the bytes in empty disk "bins" for future retrieval. At the lowest level, DOS uses one sector as one bin. Just as in a warehouse, if the number of bytes of data (quantity of an item) in the file (shipment) is greater than one data sector (storage bin) can hold, another sector (storage bin) is used for additional storage. If fact, some files could require hundreds of sectors as data storage bins.

DOS doesn't use every sector on the disk for storing files, however. DOS reserves a small part of the disk (normally track 0 sectors) for its own bookkeeping space. This reserved amount still leaves most of the disk available for data storage. Although the exact make-up and the amount of space reserved on a disk can vary, it is never a significant amount. You will see more clearly what DOS does with the reserved area as components of the reserved area of a disk are introduced in this chapter.

Understanding Storage Allocation Logging

To keep track of which sectors it has used, DOS keeps a log of all available sectors. In a warehouse, the warehouse manager might keep in a log book a record of all of the warehouse's bin locations. The manager can then consult the log book when allocating bins to a newly arrived product that needs to be warehoused. DOS uses the log of a disk's sectors to allocate storage space for new file data on the disk. Consulting the log book to determine which bins are available saves the warehouse manager a tedious search for empty bins in the warehouse. Consulting the sector log saves DOS from searching the disk for sectors which are available for allocation.

Figure 2.9 illustrates an simplified allocation log. Because of its role in storage allocation, DOS's sector log is called the *FAT* (*File Allocation Table*). Each disk formatted by DOS has at least one FAT stored in the reserved area of the disk. DOS provides for a backup FAT in the reserved area also, but its inclusion is not a fully supported feature of DOS.

Cue:
Each sector unit is called a cluster.

Depending on the disk type, DOS assigns sectors individually, in adjacent pairs, or in units of four, eight, or more sectors per unit. The FAT lists the storage locations of the disk in these sector units, rather than in individual sectors. Whether there is one sector or more than one sector in one of these units, each unit is called a *cluster*. DOS always uses adjacent sectors to make a cluster. The number of sectors in a cluster will always be an even number because DOS uses cluster sizes based on powers of 2. Table 2.3 lists the common cluster sizes for disks.

WAREHOUSE BIN Allocation Log	
BIN#	USED
1	☑
2	☑
3	☑
4	☑
5	☑
6	☐
7	☐
8	☐
9	☐
10	☐

Fig. 2.9

A simple allocation log for a typical warehouse.

The logic of the cluster is simple. If DOS needs to store an additional number of bytes in a file, the last cluster of the file may have space to hold these additional bytes. DOS then would not always have to allocate additional storage space every time a user added to a file. Multiple-sector clusters therefore reduce the number of FAT entries. A disk with 1,000 sectors and 2 sectors per cluster, for example, would need only 500 FAT entries.

On the other hand, a FAT with 1,000 available entries can track 8,000 sectors if each cluster has 8 sectors. Multiple-sector clusters increase the number of total sectors that a given FAT can track. If DOS must allocate space for a 2,048 byte file, DOS needs to make just one FAT entry if the cluster size is 4 sectors ($512 \times 4 = 2{,}048$). Increased cluster size makes DOS more efficient. If the space given to a FAT remains the same, more disk space can be allocated to data per FAT access when multiple-sector clusters are used.

DOS, however, does not want to group too many sectors into the established cluster. Many clusters of a typical PC-operator's disk are partially filled with bytes because of small file sizes, and the unfilled portion is wasted. In a disk with a 4-sector cluster, DOS will minimally allocate four times the 512-byte sector size or 2,048 bytes. With a 4-sector cluster, even a file consisting of 1 byte of data reduces the available disk space by 2,048 bytes. In a warehouse,

Table 2.3
Typical Disk Cluster Sizes

Disk Type	*Capacity*	*Sectors Per Cluster*
5 1/4-inch floppy	360K	2
	1.2M	1
3 1/2-inch floppy	720K	2
	1.44M	1
PC/XT hard disk	10M	8
PC/AT hard disk	30M	4

this system would be as inefficient as storing one screw in each bin. If most of the files stored on the disk were smaller than 512 bytes, one sector per cluster would be the most efficient ratio. Because users save larger files frequently or add to many small files such as memos, clusters of more than one sector are generally space-efficient.

DOS uses a coded indicator in the FAT for each cluster it allocates. Use of an indicator is similar to how a warehouse manager might place a check mark to indicate "in-use" beside a bin number in the log book. Unused bins have no check marks. With this system, the bin log is clear and simple. DOS marks the clusters in the FAT in a similar fashion, in that DOS can immediately see whether or not a cluster has been used (or defective) or whether it is free. During each search of the FAT for available clusters, DOS starts at the first cluster and searches toward the last cluster. The first available cluster in the FAT's list of all the disk's clusters is the next that DOS marks and uses for storage. This process is DOS's "first-found, next-used" cluster-search method.

In the case of a newly formatted disk, all clusters (except the first two used by DOS) will be available in order for storing files. As successive clusters are allocated, the disk begins to fill from sector to sector and from cylinder to cylinder in a uniform fashion. Cluster entries in the FAT will be marked from the start downward in a uniform fashion. All files receive as much contiguous cluster space as as they need and perhaps a little to spare when the files are smaller than the cluster size.

Cue:
Files stored in consecutive clusters are said to be contiguous.

DOS marks the cluster entries being used in the FAT in order from the top of the list and then down. When all new files have been allocated, the FAT will have an entry that is the point where DOS allocated the last file its last cluster. The remaining cluster entries in the FAT are not yet marked as being used. When files are stored in consecutive clusters in this way, they are said to be *contiguous*. Figure 2.10 demonstrates this concept. In the simplified version of the FAT shown in figure 2.10, the first three clusters belong to LETTER.1. Clusters 4 and 5 belong to MEMO.1. Clusters 6 and 7 are allocated to BUDGET.YTD. In this example, clusters 8, 9, and 10 have never been allocated.

If the next disk activity is the adding of a new file, some of the "never used" clusters will be used. This storing of a file's bytes in one contiguous block of clusters isn't always possible, however.

When a file is erased (removed from the disk), the clusters that the file occupied are available for DOS to reuse; the FAT reflects their availability. Conceptually, DOS erases the check mark for the removed file's clusters in the FAT to indicate the availability of the freed clusters. Figure 2.11 is the conceptual view of this freeing process. The left-side view of the FAT in figure 2.11 shows MEMO.1 being deleted. Clusters 4 and 5 are returned to DOS for later use. Notice in the resulting right view that BUDGET.YTD sits between two blocks of

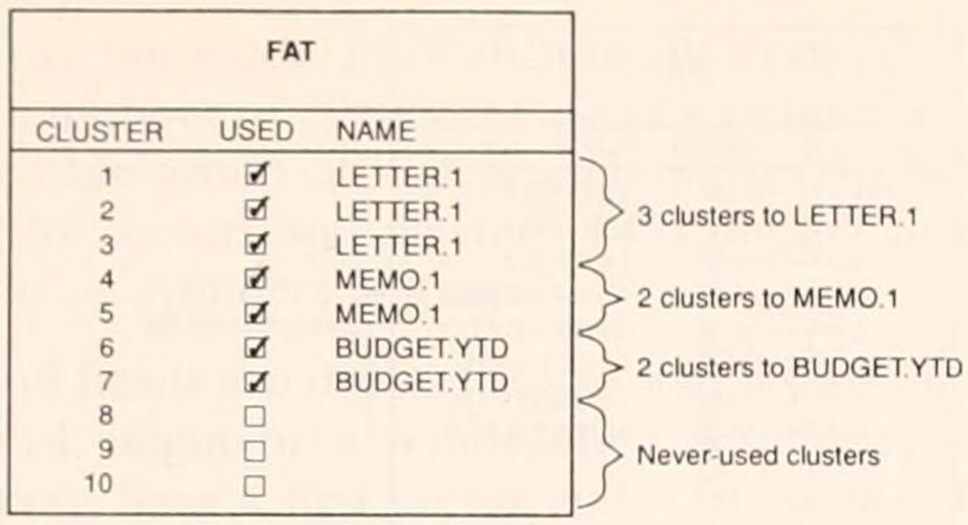


Fig. 2.10

Allocating contiguous clusters to each file in the FAT.

available cluster entries. The freed clusters are higher on the FAT's list than the empty block of clusters farther down the list where files have never been allocated.

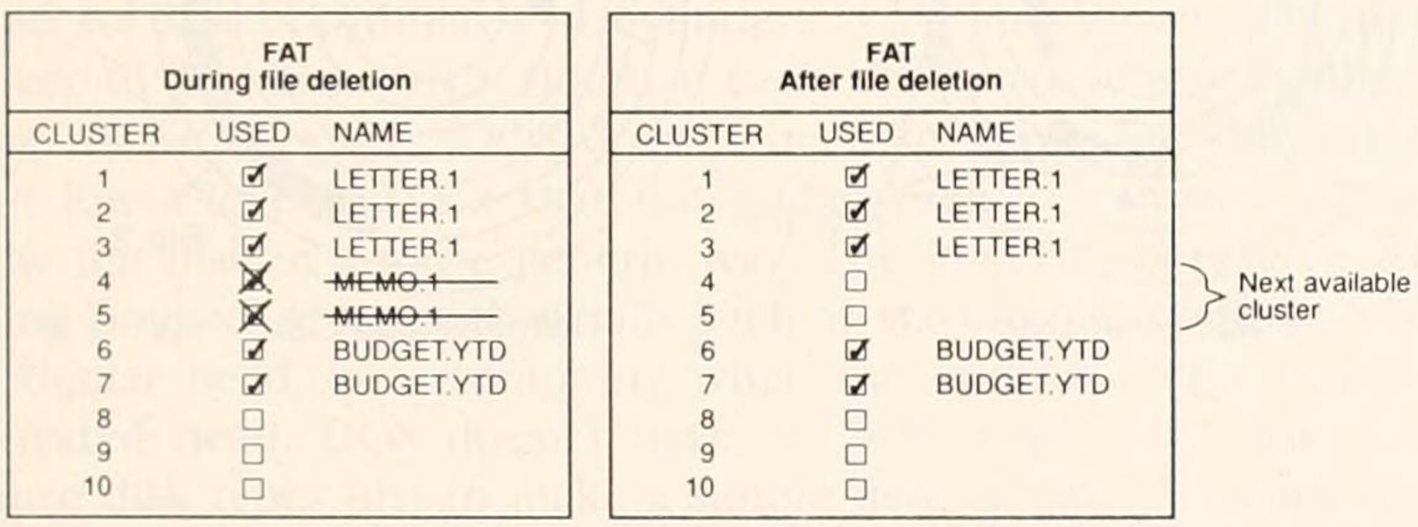


Fig. 2.11

Erasing a file and freeing clusters.

When a new file (or additions to an existing file) needs to be stored, DOS scans from the top of the list in the FAT looking for free clusters. DOS first encounters the reusable clusters and then allocates them in order to the new file. If all the freed clusters were used for the new file before all of the new file was completely allocated, DOS resumes allocating "never-used" clusters farther down the FAT again. Figure 2.12 demonstrates this situation. The file NEWFILE.ADD requires that DOS allocate 3 clusters. Clusters 4 and 5 are the first available clusters; part of NEWFILE.ADD is entered in these clusters. The remaining part of NEWFILE.ADD is recorded in cluster entry 8, the next available cluster. Now NEWFILE.ADD if fully allocated, but another file sits between the parts of NEWFILE.ADD.

Not only are the FAT entries for the new file noncontiguous, but the sectors of the storage area also are noncontiguous. The file and its FAT entries exist in chunks or fragments. By itself, allocating a file in fragments doesn't pose a problem for DOS, but actual file storage isn't always as simple a concept as keeping a group of papers tucked neatly inside of a file folder. A highly fragmented file can easily be spread out all over the disk (see fig. 2.13).

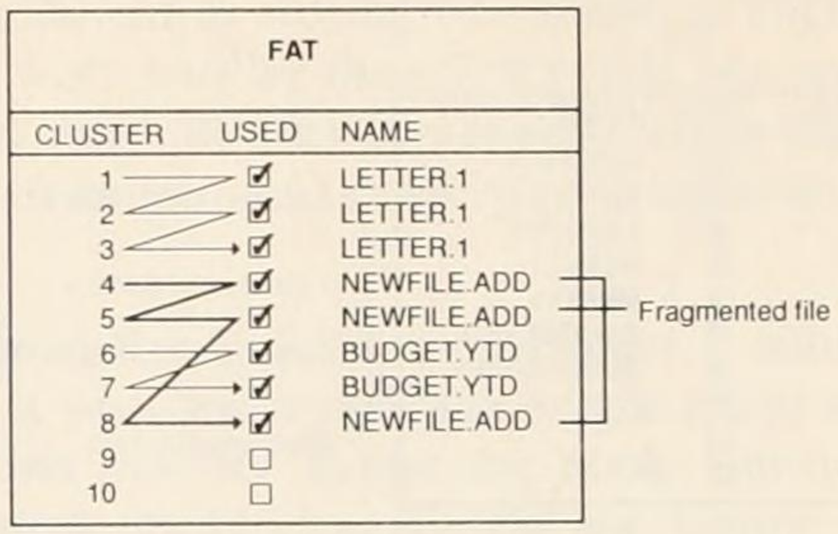


Fig. 2.12

Reallocating previously freed clusters.

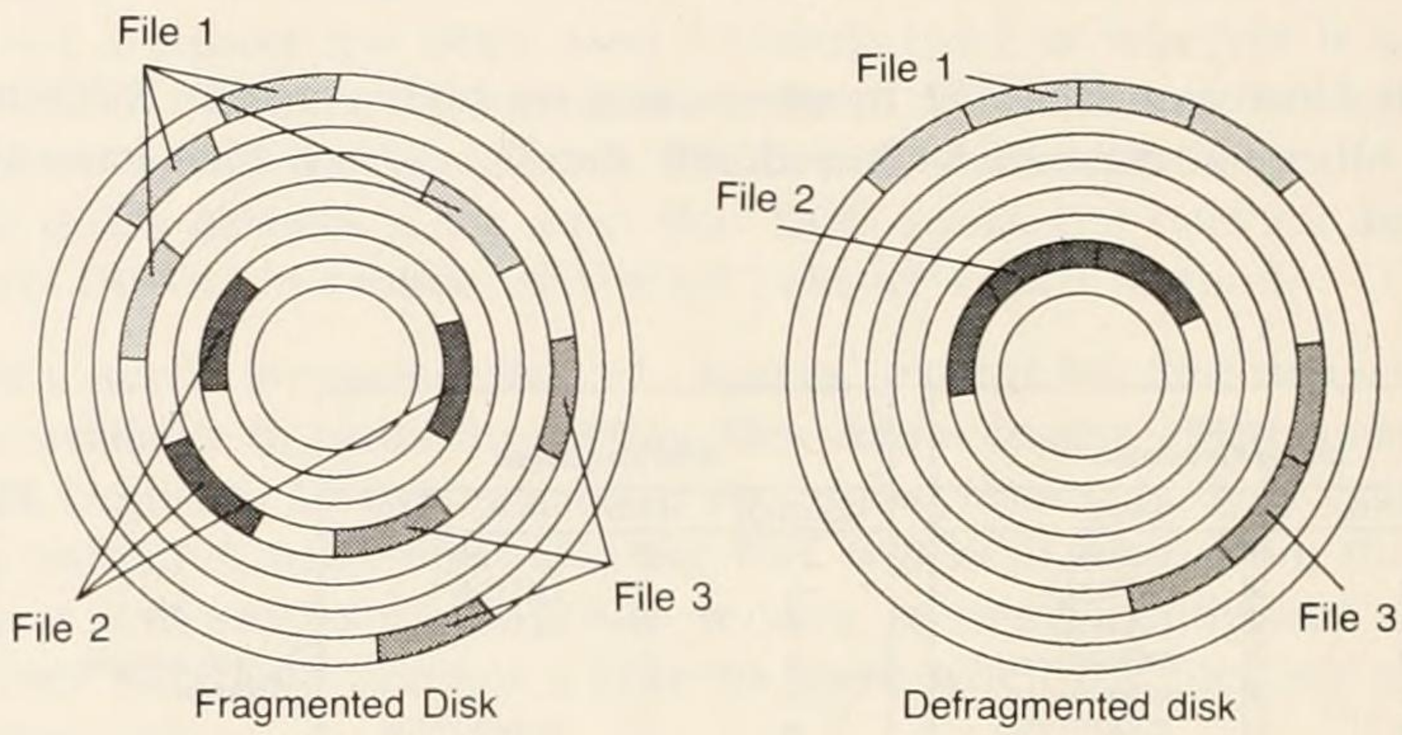


Fig. 2.13

Disk fragmentation.

Understanding Coordinates and Logical Sectors

DOS works with both fixed and floppy disk drives to locate a cluster of data by using each disk's FAT with its cluster numbers. Remember that the disk is arranged according to the head/cylinder/track coordinate system built into the formatting directions for the particular disk. A 360K floppy disk has 2 heads, 40 cylinders, and 9 sectors per track. DOS formats other types of disks differently using other formatting directions. In all cases, any given data can be located at the convergence of the desired head, at the desired cylinder, and at the desired sector.

Remember though, that disk drives have varying numbers of heads, cylinders, and sectors per cylinder. How can DOS act as an efficient warehouse manager for stored data when the disks DOS manages are so diverse in their underlying design? DOS doesn't try to track disk data by coordinates. DOS views a disk at a higher level.

The warehouse manager uses a bin log with bin numbers instead of aisle, shelf, and bin coordinates. He uses the numbered bins because all warehouses finally

come down to bin space. DOS disks finally come down to clusters. DOS "sees" all disks as a number of clusters made of a number of sectors. To DOS, the disk is an addressable group of logical sectors. As long as DOS knows the ordering method of the sectors and clusters, DOS doesn't need to log storage information in terms of cylinders and heads.

Cylinders and heads are disk-drive specific; sector numbers are general. A DOS-formatted disk might be a removable 160K floppy or a 80M hard disk. Both disks, however, have a first sector and a last sector. By numbering the sectors from one and counting the sectors cylinder-by-cylinder and from head-to-head within cylinders, DOS can use the same basic space-allocation strategy without regard to the disk's physical attributes.

The low-level driver routines are programmed to deal with disk specifics. DOS relies on the low-level routines in the BIOS layer to interface directly with the disk drive's electronics. DOS merely communicates file needs to the driver. At the point where DOS and the driver communicate, cluster numbers are translated to disk coordinates of cylinder, head, and sector and passed on by the driver to the disk drive. Because each driver routine is familiar with the disk type that it supports, DOS doesn't need to know the disk's hardware details. The lower-level service that the go-between driver provides allows DOS to view the disk in a more generic way. The lower-level service keeps DOS from being bogged down with details such as starting the drive's motor, activating a particular head, or determining when the correct sector is passing under the activated head. DOS doesn't have to memorize every disk type and predict future disk types just to make a simple disk access. The same basic numbered-cluster allocation system works for DOS whether the disk is small or large.

Knowing What DOS Stores—the Directory

Cue: Every disk formatted by DOS has a directory.

So far, this discussion of DOS's storage management has focused on allocating storage space from a record kept in a file allocation table. There is one important consideration that the discussion hasn't covered. How does DOS find what it has stored? Although the use of a FAT indicates that *something* is being stored in the appointed clusters, there is no record of *what* is being stored. On the surface, DOS shows you a file as a named collection of data. DOS doesn't record the file's name within the file's disk storage area, so it must be stored somewhere else. DOS needs some convenient means to tie a file name to where it can be found quickly on the disk. Most disk operating systems incorporate some sort of catalog or directory that indicates what is being stored as well as storage location(s). Every disk formatted by DOS has a directory.

The directory is stored in a portion of the disk's reserved space in the sectors in the first track. Each file stored in the directory is listed once by name. The DOS file directory includes provisions for other information about each file. In addi-

tion to the file's name, the total number of bytes of storage space occupied by the file is listed. Also listed is some status information such as the time and date of creation or last modification of the file.

Table 2.4 summarizes some important inclusions in DOS directories. This directory information is handy, but the directory seems to have no way to indicate which of the disk's clusters holds a file. DOS, however, provides an even better method. The FAT and a directory are linked so that they can work together to locate named files.

Table 2.4
The Main Features of DOS Directories

Feature	*What is Stored*	*Example*
File name	Eight-character file prefix	`THANKYOU`
File-name extension	Three-character file suffix	`THANKYOU.OLD`
Time	The time of creation or last modification	`10:22`
Date	The date of creation or last modification	`11-14-89`
Starting cluster	The number of the first cluster allocated to this file by DOS in the FAT	`576`
File Attributes	Special status information about this file used by DOS.	`Read-only/ Hidden`

You will recall that the FAT contains a list of cluster numbers, one for each cluster on the disk. In a simplistic view of a FAT, you might picture that there is a box beside each cluster number. The resulting FAT would be very similar to the one represented in figure 2.9. If the cluster is in use, the box is check-marked; if the cluster is available, the box is empty (not check-marked). DOS can allocate clusters with this FAT because DOS can easily determine the first free cluster by looking for an unmarked box. Now, if you replace the box with an entry blank beside the cluster, DOS can enter a variety of information in the blank. DOS, for example, can enter a binary value represented here by the word "free" if the cluster can be allocated. The meaning signified by "free" is the same as the meaning signified by the absence of check mark in a box in figure 2.9.

Figure 2.14 shows the visualization of the improved FAT with its entry blank as a link status. Each cluster in the FAT has an entry for a link status. When a

cluster logs part of a file, the link status shows the cluster number of the next cluster that is allocated to the file. Clusters that are available because they are no longer part of a linked list of clusters (in this case, clusters 4 and 5) are "free" clusters. You can presume that clusters 4 and 5 once were allocated to a file which has since been erased. Clusters that have never been part of a linked list are also "free." In figure 2.14, clusters 9, 10, and those that would follow in the complete FAT, have never been allocated. "Free" clusters are not part of a linked list of clusters belonging to a file. DOS is responsible for marking a usable FAT entry as being "free."

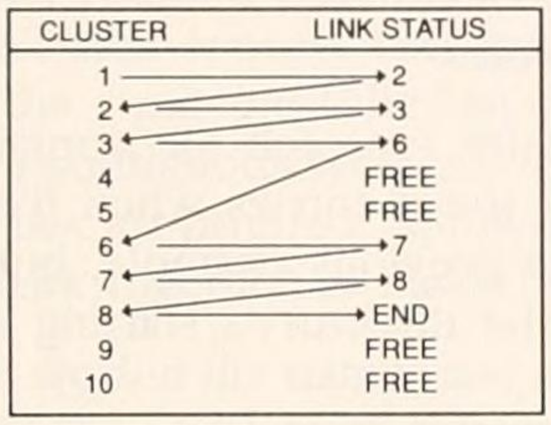


Fig. 2.14

The concept of linked clusters in DOS's FAT.

DOS has two ways of using the entry in the FAT to indicate that the cluster is part of a file. The first way is by placing the linking number to the next cluster used for the file in a FAT entry. The other way is to put a value that indicates the "end" in the file's last cluster entry. Not only do the linking number and the "end" entries indicate that the clusters are unavailable for allocation, they also form a traceable chain for each file allocated.

In practice, DOS allocates a cluster to a file and then determines whether the entire file will fit into the cluster. If the entire file will not fit, DOS stores as much as it can in the cluster. DOS then finds and allocates another cluster. When the next cluster is found, DOS puts the number of the second cluster back in the entry blank of the previous cluster. DOS has chained the two clusters together by using the entry in the first to point to the location of the second. If the remaining bytes of the file do not fit in the second cluster, DOS again stores what it can and finds another cluster. DOS will repeat this partial store-and-allocate process until the entire file is allocated to clusters. Each cluster except the last has the number of the next cluster as its entry. Because the last cluster is the end of the file's chained allocations, the last cluster has a value of "end" in its entry.

The example in figure 2.14 shows the linkage of clusters 1-3 and 6-8 for a file allocated by DOS. You can assume that clusters 4 and 5 were allocated to some other file when the rest of the clusters were allocated. Even with the fragmentation of the file between two blocks of clusters, the entry in cluster 3 which points to cluster 6 as the next cluster leaves DOS a trail.

Using this "point-to-the-next-cluster" method, the contents of the blank for any cluster number in the FAT is either free, end, or the number of the cluster to examine to find additional bytes of the file. One file's sequence of FAT entries forms a chain of cluster numbers; any link of the chain either ties to another link (cluster) or is the end link in the chain (the file's last cluster). Utilizing this chain, DOS can start at the first cluster number entry for a file in the FAT and quickly trace that file's other clusters through the chain to the end cluster. DOS knows where all of the bytes of the file were stored. This use of the FAT is a sort of scavenger hunt with each clue giving the exact location of the next clue. DOS only needs to know where the first entry for a file is located in the FAT, and it can access the entire file. DOS needs a link between the FAT's "where" values and the directory's file names.

Remember that the directory entry for each file contains the starting cluster number for the file. DOS makes these entries when files are allocated. Figure 2.15 shows the same FAT as the previous example, but also shows the directory's link to the FAT through the directory's starting cluster number for the file.

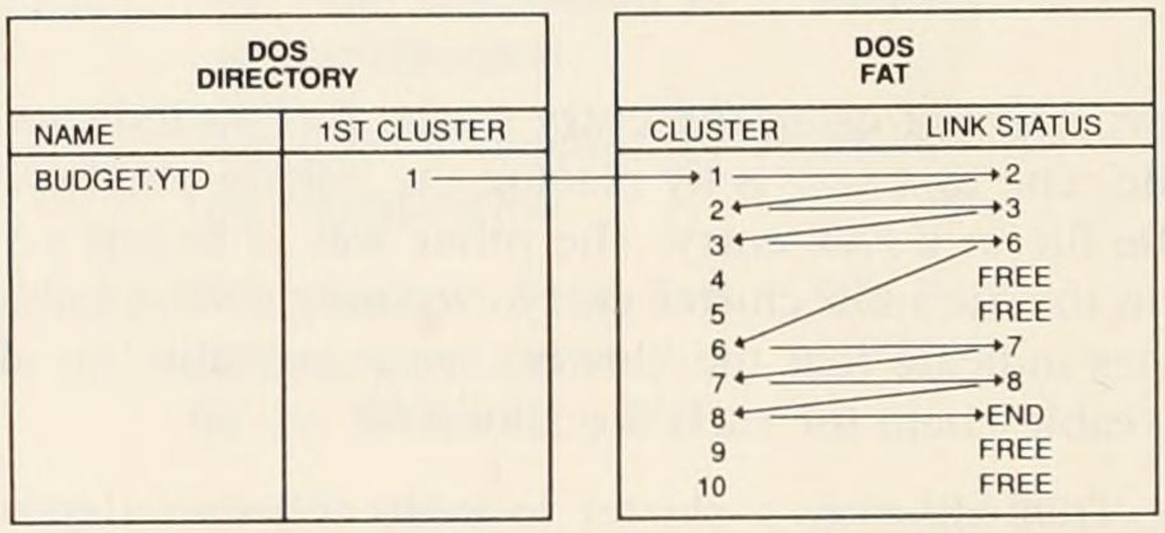


Fig. 2.15

The directory linked to the FAT by starting cluster.

DOS enters the directory information for BUDGET.YTD and then allocates the first available cluster in the FAT to the file. DOS puts that first cluster number in the directory for future access. DOS allocates the rest of the space for the file and chains the allocated clusters together in the FAT. DOS's method for file allocation and access is the same for all files.

With this system, DOS allocates free space quickly and finds all the bytes of a file regardless of where the file is stored by using the FAT. The file name, file status, and the pointer to the first entry in the FAT are available for each file with the directory.

Cue:
Each file stored on a disk is entered in the directory table by DOS.

Now apply this FAT-plus-directory system to DOS file management. Every disk formatted by DOS has some space reserved in the first track of the disk to serve as a directory table. DOS also uses some of this space to create a FAT that matches the number of clusters on the disk. Each file stored on a disk is entered in the directory table by DOS. When a new file is added to the disk, DOS con-

sults the FAT to find the first cluster that can store the file and puts the first cluster number in a field in the directory entry for that file.

As clusters are allocated to store the file, the current cluster number is entered into the preceding cluster's pointer entry. If the file has been completely stored on the disk, the final cluster entry for that file in the FAT is marked with a value DOS takes as "end." When DOS gets a request for a file (such as from a COPY command), DOS looks for the file's name in the directory. When DOS finds the name, it uses the first cluster entry to directly access that cluster's entry in the FAT. DOS calls for low-level service from the BIOS routines to read the cluster's data at the cylinder, head, and sector coordinate that the disk understands. If the cluster entry in the FAT points to another cluster, DOS consults that other cluster's entry in the FAT and repeats the read steps. DOS will continue to move to each cluster in the chain until the last cluster's data is retrieved. If the file is fragmented (in non-contiguous sectors), the disk drive will have to work a little harder, but DOS has no problem following the file's cluster chain. The directory and the FAT work together to make DOS more efficient.

The directory is good for storing file names and associated status information. If the disk capacity is small, the number of entries in the directory will be small because the directory's "living quarters" in the reserved area of the disk is limited. The dedicated reserved space for the directory will accommodate all of the limited number of entries; searching the entries for a file name is therefore reasonably fast. If the disk capacity is huge, however, the number of files the disk can accommodate is proportionally huge. The space reserved by DOS for the directory on a large capacity disk might become filled before the disk becomes filled.

DOS designers faced this limited directory problem in V1 when hard disks began to catch on with users. The DOS designers facing the larger disks solved the directory limitation problem. But, before you see how they solved the problem, you should see how they could have solved the problem and why some solutions were not implemented.

Enlarging the reserved space would allow more directory entries, but establishing how much directory space is enough is an arbitrary judgment call. If DOS designers provided for 5,000 files to be stored in the directory, those users who needed 10,000 files would be disappointed. If the designers provided for 10,000 entries in the directory, those users who needed only 1,000 entries would be sacrificing usable disk space to hold a directory that couldn't be fully utilized.

Unless some cap were established on the number of entries, the directory could take the lion's share of the disk. Besides, because DOS makes directory entries on a first-come, first-serve basis, file names will not be in any particularly useful order in the directory. To find a file, DOS has to search from the start of the directory until it finds the name it is looking for. This long search would be cumbersome with a filled, over-sized, disk directory.

In addition, there wouldn't be a way to instruct DOS to divide files into categories by their purpose, what programs normally use them, what special functions they support, or other categories. Searching a category is much faster and to the point than searching every entry in a large directory.

Considering the disadvantages of simply expanding the directory size to accommodate more files on a hard disk, DOS designers had to answer one question: How can you use the same basic directory system and gain the benefits of categories of your choosing without having a confusing, slow-to-use directory that might fill with file entries long before the disk filled with files? The DOS designers answered this question in V2.0 by introducing hierarchical directories.

Understanding the Root Directory and its Subdirectories

To maintain the functionality of a directory-based system, DOS designers designed one master directory with a predetermined number of entries to keep the reserved space small. The designers first established a cap on the number of entries in this master directory to fix its size. The cap number was proportional to the capacity of the disk the designers were dealing with. Floppies had fewer file entries in the master directory (less bytes) than hard disks.

The master directory was not intended to list the names of every file on the disk (unless the disk was a floppy used to store a few files). The master directory, as the *root* of the system, was called the *root directory*. Many of the disk's files could be entered into subcategory expansion directories or *subdirectories*.

Ideally, the subdirectory system would not limit the number of files it could manage. Like a single directory system, the subdirectory system was efficient and easy to use. This subdirectory system had the root directory as its only required directory and optional subdirectories subordinate to the root.

An earlier mainframe operating system, UNIX, solved the directory problem by allowing users to create their own subdirectories as expansions of the main directory. Borrowing this UNIX-like directory system, DOS incorporated subdirectories into its file system.

In the root directory or in a subdirectory, DOS can still enter the name and the first cluster number of a file's FAT entry. The DOS subdirectory is a special file that DOS uses like the root directory. On the surface, you can't tell that the root directory is a reserved section of the disk, and that the subdirectory is a special file. DOS manages the root directory and subdirectory difference internally and provides uniformity on the surface. For this reason, it is quite common for DOS users to call subdirectories "directories." Unless the context of the term dictates otherwise, subdirectories can be called directories.

DOS can implement an entire system of subdirectories and still retain the general advantages of a single directory. From a space-economy point of view, a subdirectory system overcomes many of the drawbacks of a single directory system, such as an excessively large number of file entries. Of course, DOS has to provide a few commands to manage the subdirectory system. Commands to create, remove, and change the current directory are examples.

On a freshly formatted disk, you have access to the disk's only directory, the root directory. With the MKDIR (Make Directory) command, you can create a subdirectory called LETTERS. You can use the name LETTERS as the subdirectory name to categorize the letter files that you will keep in this new subdirectory. To focus DOS's attention on the LETTERS subdirectory as though it were the only directory on the disk, you can use the CHDIR (Change Directory) command to change the current directory to LETTERS. If you later want to remove the LETTERS subdirectory because it is no longer needed, you can use the RMDIR (Remove Directory) command to do so. DOS's subdirectories, as well as the commands that support them, are a great advancement over the single fixed directories of V1.

Subdirectories as Files

Cue:
A subdirectory can be considered DOS's personal file.

DOS can utilize subdirectories while maintaining a fixed size reserve area on a disk because DOS is in charge of files. DOS makes a subdirectory out of a file. A subdirectory can be considered DOS's personal file.

A subdirectory is not placed in the reserved area like the root directory. Instead, it is located in the main storage area of the disk just like other files. Unlike the master directory, which is fixed in size, a subdirectory can grow as more entries are added to it. All subdirectories have names similar to files, and to DOS, a subdirectory's name is a modified file name. Because there is only one root directory and its space is reserved, it has no recorded name and is always known as the root. Naming subdirectories gives you the ability to categorize the contents of a subdirectory by the meaning of its name. DOS attaches no special meaning on your choice of a subdirectory's name. Internally, DOS can search for a subdirectory name in a manner similar to the way it searches for a file name.

A Hierarchy of Directories

The first subdirectory you add can be considered an "offspring" of the root directory. You create it from the root. It is the child of its parent (root) directory. Your first subdirectory is not, however, the end of the parent-child relationship. Subdirectories can contain other subdirectories as child directories. A child-expansion directory is subordinate (lower in the ordering) to its parent directory in the file system. Figure 2.16 shows a typical hierarchy of the DOS file system.

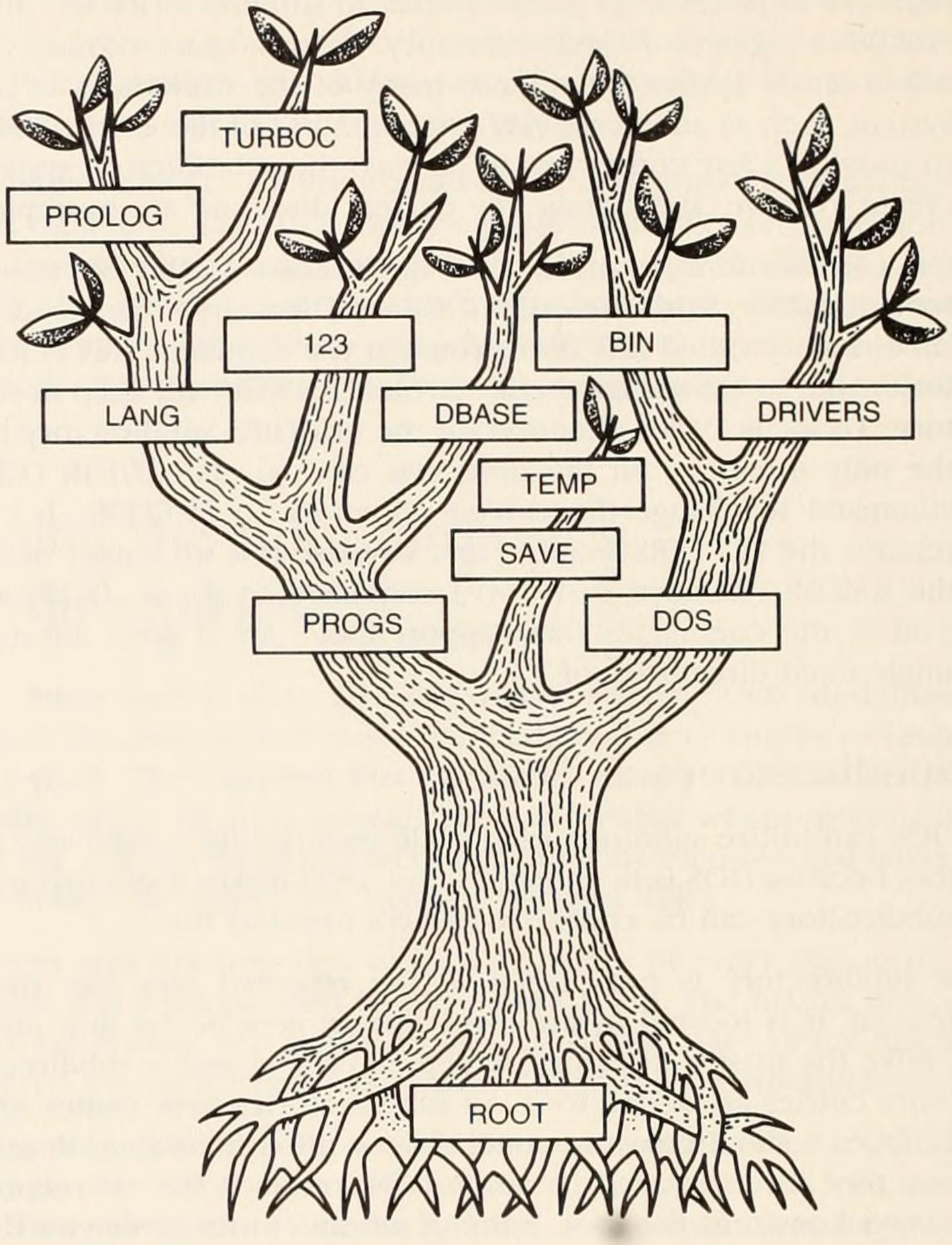


Fig. 2.16

The DOS tree-structured directory system.

Cue:
The directory system of DOS is organized in a hierarchical fashion.

When viewed from the parent-child perspective, the typical DOS file system resembles a family tree with the root directory branching through generations of subdirectories with files as leaves coming from each branch (see fig. 2.16). There is a natural hierarchical ordering to this type of tree structure. Using this hierarchical ordering approach to file-system management, DOS allows you to specify a directory structure as elaborate as necessary.

Yet DOS can always keep file names and their physical disk locations in order. When you tell DOS to create a directory, DOS handles the details of making a new file and marking it as a directory. DOS also makes an entry in the new directory for the starting cluster of the parent directory and another entry for the starting cluster of the new directory itself. These two directory entries in the newly created subdirectory are useful to DOS when DOS needs to find its

way back to a parent or grandparent directory. The DOS designation for the parent directory entry is called .. (dot, dot), and the designation for the directory itself is called . (dot). Because the root is the top level of the hierarchy and has no parent, the root has no . or .. entries. The hierarchical directory structure of the DOS file system provides large storage capacity, subdirectory privacy, and efficiency for DOS's disk related services.

Directory Paths

When you want to access files in a particular subdirectory, you can give the full path name down the directory hierarchy, starting at the root. The path name consists of the \ (backslash) specifier for the root; backslashes also separate subdirectory names that trace the family tree to the files. The file name is the eight-character name plus an optional period and a three-character extension. In the LETTERS directory in a previous example, the full path name of the MOM.LET file is \LETTERS\MOM.LET.

A full path name is the path name plus the file name and extension. Every file in the file system has a full path name, even if the file belongs to the root directory. The file CONFIG.SYS in the root directory has a full path name of \CONFIG.SYS. No full path name can appear more than once in a disk's file system. The full path name is DOS's positive identifier for a file. A duplicate full path name would eliminate this positive ID method.

Reminder:
A full path name is the path name plus the file name and extension.

The same full file name (file name plus extension) can appear in every directory because every directory has a unique path name. The file MOM.LET can appear in \LETTERS and \MAIL at the same time. The full path names \LETTERS\MOM.LET and \MAIL\MOM.LET make the two files unique to DOS even if their contents are exactly the same.

Reminder:
No full path name can appear more than once in a disk's file system.

DOS always considers the path name when doing its internal bookkeeping. Because each disk has its own file system (and therefore bookkeeping provision), file and path names can be duplicated from one disk to another. The files A:\LETTERS\MOM.LET and C:\LETTERS\MOM.LET are two distinct files in DOS's view even though they both have a full path name of \LETTERS\MOM.LET.

DOS keeps this duplication straight by relying on the drive name (the letter such as C:) to distinguish your intent.

Understanding Defaults

To make entering commands more efficient, DOS provides for predetermined values, *defaults*, to be used when you omit certain parts of a drive and path name in a command. The default disk drive is established by the boot process

and automatically becomes the drive that your PC boots from. The default directory is automatically set to the root directory of the boot disk.

Cue:
Changing to a new disk or directory makes it the current or default drive or directory.

You can change the default drive and directory to the drive and directory that contain your desired files. Changing to a new disk or directory makes it the *current* drive or directory. When you boot your computer from drive A:, the current drive is A: and the current directory is \. If you enter **C:** at the DOS prompt, you current drive is then C:. If you then enter **CD \LETTERS**, your current drive and directory become C:\LETTERS.

A current drive or directory is the same as the default drive or directory. DOS assumes that files are accessible on the default drive in the default directory unless you use a drive name and a path name in a command.

If you don't clearly understand the concept of using defaults in commands, don't worry. Part II of this book will explain defaults in detail. For now, just understand that through defaults, DOS can accept commands that treat a disk and a directory as if it were the only disk and directory in the file system. The commands that use DOS defaults look very simple considering the potentially complicated hierarchical file system that DOS is tracking below the surface.

Understanding Disk Partitions

The growing storage capacity of hard disks offers an opportunity for people to use their PCs as a hardware platform for operating systems in addition to DOS. Many hard disks today have the storage capacity to support the files of more than one operating system. Without some provision to host more than one operating system on the same hard disk, however, a PC needs a different hard disk for each operating system.

Cue:
A partition is a subsection of a disk.

Luckily, such a provision is available. It is the *disk partition table*. The disk partition table is located at the very start of a hard disk's sectors where it is available for any operating system to examine. The disk partition table has entries to log up to four partitions. Each partition is a number of cylinder and head units of the hard disk. In other words, a partition is a subsection of the disk. Of course, if the disk is given only one partition, the entire disk is devoted to the operating system that controls that partition.

IBM introduced the concept of a partitionable disk with PC DOS V2.0. The provision for partitions was introduced to coincide with the introduction of the IBM PC XT computer. Every version of DOS introduced since V2.0 has included provisions for partitions.

The hard disk partition table is located in the track(s) of the disk reserved for operating system tables. DOS can't really claim to own the partition table as one of its bookkeeping features. The partition table is a hard-disk level bookkeeping

provision which all operating systems have equal access to. The partition table indicates which cylinders are controlled by which operating system.

QNX, a UNIX-like PC operating system, can access the partition table of an IBM PC or compatible's hard disk. When you "mount" the hard disk's QNX partition, QNX accesses the disk partition table and scans the entries to find the first QNX partition. Other PC operating systems access partition table data in a way similar to DOS and QNX. The XENIX operating system also honors the disk partition table's entries.

Two or more tenants who share a warehouse facility might draw a yellow line down an agreed line to divide the warehouse space. On a hard disk, the disk partition table formalizes this type of imaginary line. The physical space of the hard disk can be divided into partitions. Figure 2.17 will help you visualize this division.

Partition 1 in figure 2.17 is on the outer cylinders of the disk. Partition 2 begins at a cylinder boundary and extends to the centermost cylinder of the disk. Although one disk is present, the two partitions defined in the partition of the disk act as two.

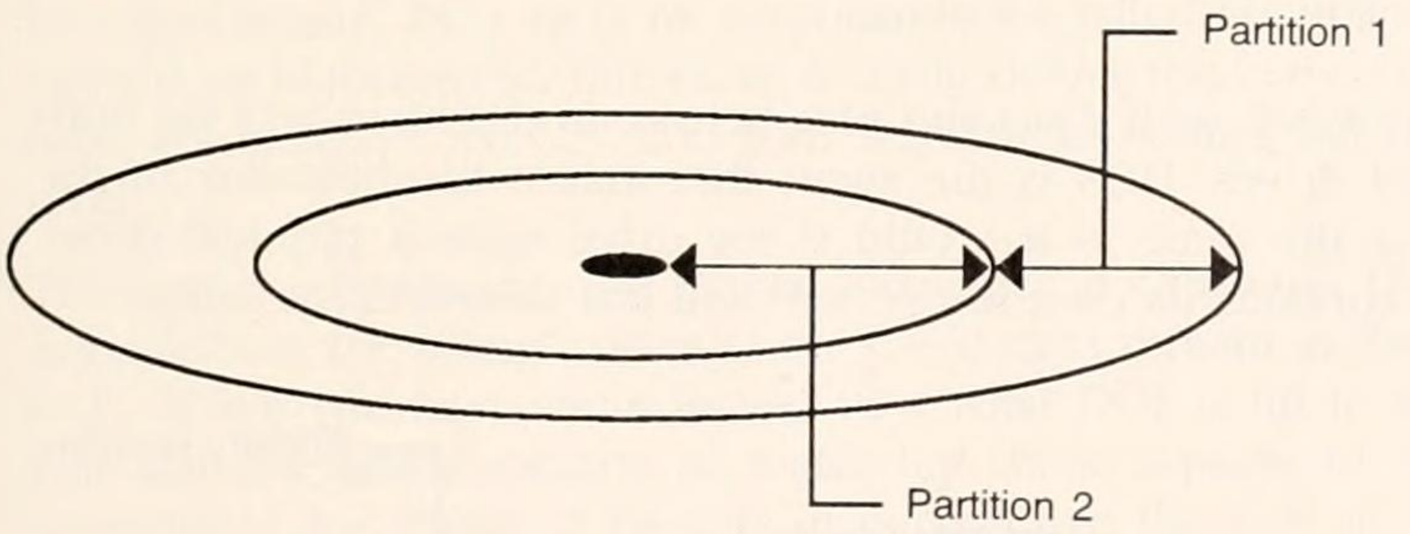


Fig. 2.17

The physical division of a single disk through partitions.

Unlike the warehouse example, however, only one operating system can be active at a time. The division of the hard disk, however, is still respected by the active operating system. The active operating system will confine its file-management activities to its partition. Each operating system has a utility to access and maintain its part of the partition table. DOS uses the FDISK utility to manage its partition definitions. QNX, as well as other operating systems, also names its partition-table maintenance command FDISK.

Reminder:
Only one operating system can be active on a disk at any one time.

When DOS gets its starting and ending cylinders from the partition table, it uses the starting cylinder of the partition as though it were the first cylinder on a physical disk. The FAT and root directory tables, as well as other DOS-reserved information, start in the first cylinder. QNX lays out its partition as though the partition were the entire disk. Internally, QNX uses the first cylinder of its parti-

tion as though that cylinder were the first cylinder of the disk. QNX can also read DOS's partition table information and mount the DOS partition to the QNX file system. With a special command, you can convert DOS files to QNX files.

Although you might not ever use the flexibility that the partition table offers in multioperating system operation, the partition table is there on the hard disk should you need it.

Cue: DOS V3.3 introduced the capability of the extended partition. The extended DOS partition can be one or more logical drives.

A hard disk's flexibility is further enhanced when this same partitioning concept is extended so that DOS can manage more than one partition. Starting with V3.3, this extended partition capability was included with DOS. The physical hard disk can include a standard or *primary* DOS partition as well as an *extended* DOS partition. The extended DOS partition can be one or more logical drives. The concept of logical drives is seen in figure 2.18. Logical drives are not real disk drives; rather, they are defined sections of a real disk drive.

In figure 2.18, the outer cylinders of a hard disk have been allocated to the primary DOS partition. The extended DOS partition begins after the last cylinder of the primary partition. The extended partition extends to the last cylinder of the disk. Notice the two dotted circles further dividing the secondary DOS partition. Each of the dotted circles marks the beginning of a logical drive. In this example, drives C:, D:, E:, and F: have been created through the partition table.

The commands that work with disks and files behave in the same way on both logical and physical drives. DOS is the agent that makes the behavior of the logical drive appear the same as it would if the drive were a physical drive. Each logical drive contains its own file system and has its own disk name.

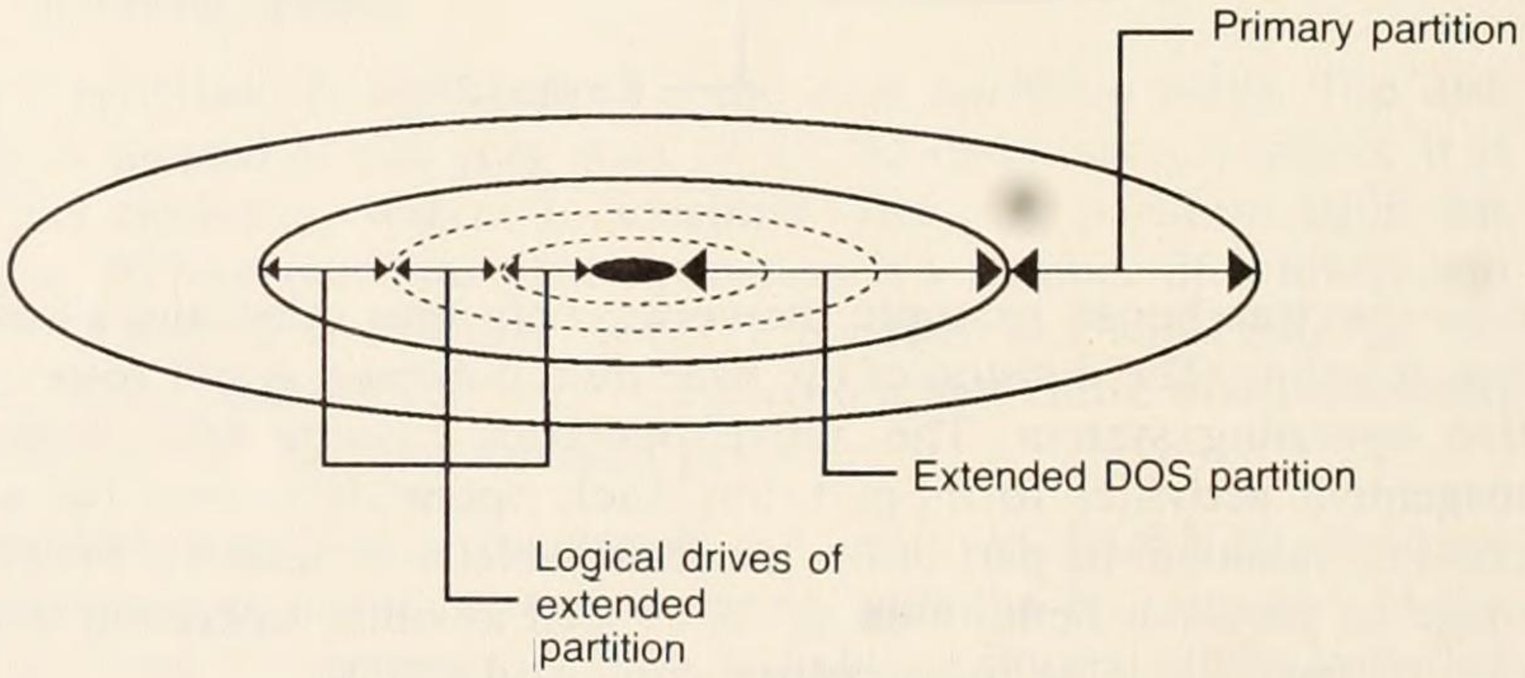


Fig. 2.18

An extended partition with logical drives.

The extended DOS partition is very similar to a primary or standard DOS partition. The big difference is that DOS cannot boot from an extended partition. The FDISK partition-maintenance command provided by DOS has no provision to designate the extended partition (or its logical drives) as a bootable parti-

tion. Other operating system partitions, such as a QNX partition, are not affected by DOS's extended partition.

Only the primary DOS partition can be bootable. At the command-issuing level of DOS, there is no practical difference between physical and logical drives. Files stored on a logical drive D: appear to you as though they were on their own drive, but they are on the same physical hard disk as the files on logical drive E:. Each logical disk has no log of what is stored on another disk or logical disk. From DOS's bookkeeping point of view, logical drives are totally separate entities.

Reminder:
Only the primary DOS partition can be bootable.

Looking at DOS's File System from the Ground Up

As you can see, the DOS file system and its related mechanics aren't as simple as they appear on the surface. At the DOS command prompt, you issue commands and rightfully don't expect to be responsible for the success or failure of the command's execution. Command execution is the responsibility of DOS. Needless to say, DOS can't be responsible for failed commands that are poorly formed or inaccessible. But there is a big difference between your learning to issue commands correctly, and your learning to control hardware at the binary level.

Through the hardware, the driver software, and through DOS's own internal organization, the complexities of the file system remain in the capable hands of DOS. When you have some insight into what DOS is up to behind the scenes, you stand a better chance of mastering those aspects of DOS that you are responsible for. Figure 2.19 puts in perspective the role of DOS in disk-based activity. The figure shows a series of transformations or refinements of organization that become the structure for your disk and file work.

Figure 2.19 shows boxes containing a level of data organization. Connecting the boxes are a series of arrows. Each arrowhead references the method used to transform the level of data organization from the preceding box to the next box. Follow the transformation of magnetic pulses and fields through to the multidisk file system that DOS offers.

From magnetic pulses, the drive head extracts the electrical signals that the drive electronics can transform into electrical binary data. The drive electronics, in cooperation with the disk-driver program, package these binary signals into bytes and deliver them to the PC's memory in 512-byte sectors. DOS's format-defined clusters keep the bookkeeping of the sectors efficient. The sectors give rise to tracks; tracks give rise to cylinders.

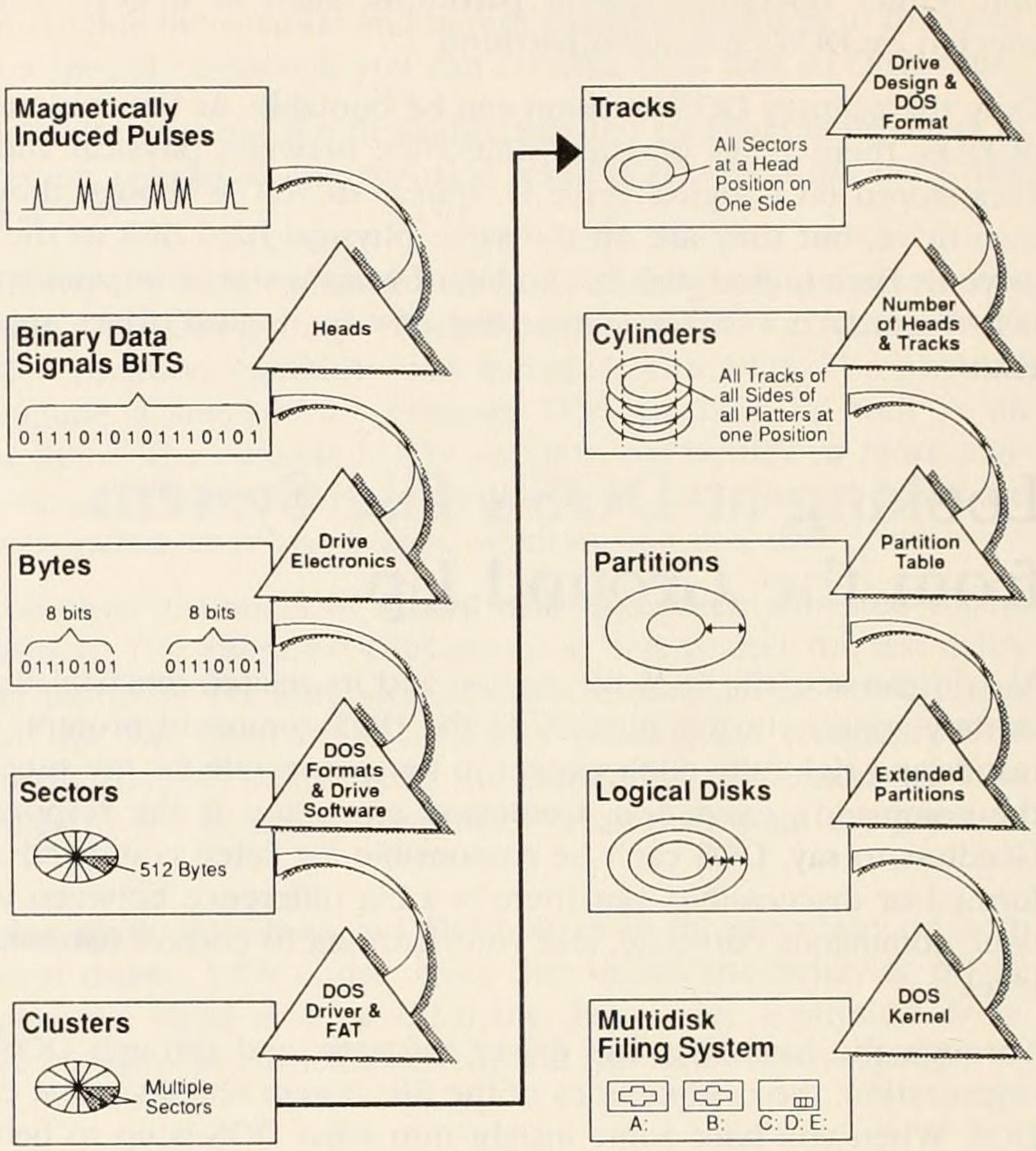


Fig. 2.19

The underlying organization of the DOS file system.

DOS keeps a logical map of this disk real estate in the FAT and matches file names to their location on the disk through the directory. DOS knows a disk's organization as soon as it reads the disk's boot record. DOS transparently treats one hard disk as many disks when DOS finds the disk's division into partitions in the partition table. DOS uses names for both physical and logical drives so that you can manipulate a multidisk filing system using hierarchical directories. When you consider that DOS manages this range of details from magnetic pulses to multidisk filing systems, is is hard to believe that DOS has its "disk act together" in the moment it takes the drive light to glow.

Summary

In this chapter, you learned some of the fundamentals of the DOS file system. Following are the key points to remember:

- DOS tracks named collections of data in files.
- Disk drives are the principal file storage devices for DOS.
- Disk drives appear logically the same to DOS because DOS accesses disk drives through the services of drivers.
- DOS provides formats that use the capability of a disk drive to produce a number of tracks for each of a number of heads; each track is then divided into a number of sectors.
- Although the DOS formats are usually applied to specific disk sizes, the formats are logical and can be applied to different disk sizes.
- DOS allocates disk space to files in cluster units. Clusters are logged in the File Allocation Table or FAT.
- DOS tracks file names in the directory. Each DOS-formatted disk has one primary or root directory. All DOS directories link a file's starting cluster number to the FAT.
- Because each disk formatted by DOS has its own FAT and root directory, each DOS disk has its own file system.
- DOS V2.0 introduced subdirectories to the DOS file system. You manage subdirectories by using DOS commands.
- DOS subdirectories allow the file system to be arranged in a tree-like hierarchical structure of your design.
- The FDISK command manages DOS's entry in the disk-level partition table.
- DOS V3.3 introduced the extended DOS partition. FDISK can divide the extended partition into additional logical drives.

By understanding the concepts presented in this chapter, you will be better equipped to understand the commands that DOS provides to manage disks, directories, files, and the file system. Chapter 3 will introduce the DOS command processor, COMMAND.COM, and the organized computer environment in which it operates.

3

Understanding the Command Processor

Cue:
The DOS command processor is a program called COMMAND.COM.

The most visible part of DOS is its command processor, which is a program named COMMAND.COM. At the DOS level, the command processor presents the familiar C› prompt and waits for your keyboard command. Behind the scenes, the command processor is prepared to handle the critical interrupts generated by your PC when processing is in jeopardy, such as when a disk file is unreadable. Commands interpreted by the command processor may be DOS commands or the names of programs. In either case, the command processor oversees the execution of the command by your computer.

The computer is the hardware forum that the command processor uses. At the instant that you turn on your PC, it cannot process DOS commands. In fact, the steps that a PC goes through before DOS commands can be processed are numerous. The hardware forum must be prepared and the PC's resources must be assessed and brought under control before the command processor can do its job.

This chapter provides descriptions of the boot process and COMMAND.COM's control of the PC. To make COMMAND.COM's role more clear, the chapter discusses the computer's hardware and software and the invisible layers of an operating PC. You will learn about syntax, parameters, and switches and understand the difference between an internal DOS command and an external DOS command. Through the discussion of COMMAND.COM, you will gain a more thorough understanding of the role of the command processor in DOS.

Examining the Role of Hardware

A PC's hardware is the collection of electronic and electromechanical components composing the signal-level layer of the PC. The hardware's job is to store, transfer, reinforce, encode, decode, accommodate, and gate the binary signals

that occur during the act of computing. To many, the nature of hardware seems complex and forbidding. You may have seen a television comedy in which a naive handyman takes the cover off a neatly covered box, and springs, gears, and wires fly out of the box in chaos. Many believe that a similar fate awaits anyone who removes the case from a PC's system unit. Reading *Upgrading and Repairing PCs* (published by Que Corporation) will convince you that a useful understanding of PC hardware is within your reach. Fortunately, you don't have to disassemble your PC to see how the hardware works. The concept of hardware can be divided into two general categories: peripheral hardware and system-unit hardware.

Peripheral Hardware

Cue: *Peripheral hardware is not directly associated with a PC's system circuits.*

Peripheral hardware is hardware that is not immediately associated with a PC's system circuits. A hard disk and its controller are examples of peripheral hardware. A printer is peripheral hardware. In fact, almost any item that attaches to your PC and provides it with *input* (items such as a keyboard or mouse) or *output* (items such as a screen or printer) is an example of peripheral hardware. The act of providing input and/or output is termed *IO*. Most peripheral hardware is concerned with IO to or from the system-level hardware. Whereas you view peripheral hardware as a collection of physical devices, DOS views peripheral hardware in a more symbolic and less physical way.

DOS treats peripheral hardware as logical devices by relying on lower-level software called *device drivers* to present input and output from the hardware in a predictable, uniform way. Device drivers are a part of DOS's basic input-output system and are part of DOS's unseen BIOS extensions. Chapter 4 covers the external hardware in a conceptual way through a discussion of devices. For now, just note that peripheral hardware differs from system hardware in the computer's logical organization.

System-Unit Hardware

Cue: *The circuits and components typically on the PC's main circuit board are system-unit hardware.*

System-unit hardware consists of the circuits and components that traditionally appear on a PC's main circuit board. Newer PC designs use components that are integrated into circuit packages on a larger scale than was the case in the past. This larger-scale integration has allowed space savings that make possible putting peripheral-hardware-controller circuits next to system-unit-hardware circuits. A built-in Video Graphics Array, or VGA, circuit that provides the control of the color monitor in an IBM PS/2 or some COMPAQs is an example of a large-scale integration.

Today, function rather than physical location determines whether hardware is peripheral or system in nature. Normally, people associate the PC's micro-

processor and support components, the system memory (RAM), the ROM, and the interconnecting buses with a definition of the system-unit hardware. The main printed circuit that the system hardware components are mounted on is called the *system board*, also referred to as the *motherboard*. The next sections look at each of these parts of system hardware in more detail.

The Microprocessor

Cue:
The microprocessor is called the CPU.

The microprocessor, like memory, is an integrated circuit component, or *chip*. The microprocessor is responsible for performing logical and arithmetic operations on binary data to produce meaningful results. Simply put, the microprocessor is the chip that computes in a microcomputer. Because microprocessors play the central role in processing, they are also called *central processing units*, or CPUs.

The IBM PC and compatible PCs use the Intel 8086 family of microprocessor chips. Microprocessors compute by loading a series of instructions and executing each instruction in turn. The instructions are bytes or multiples of bytes that form a pattern of on and off signal levels (logical 0's and 1's) that are meaningful to CPUs of the same family. The binary-level instructions are called *machine instructions* or *machine code* because they exist at a level that the machine (the PC) understands. You can think of machine code as a language spoken in the CPU's native tongue. This language is called *machine language*. The microprocessor executes a program by using machine language. Even early IBM PCs can execute thousands of instructions per second through their 8088 CPUs. All CPUs execute instructions in an orderly fashion. The CPU keeps track of its current location in a series of instructions through the CPU's internal program counter.

Initially, when the CPU starts a program, the CPU's internal program counter is set to 1. The CPU then gets instruction number 1 from memory location 1, executes the instruction, adds 1 to the program counter, and gets the next instruction from the memory location that matches the new value of the program counter (in this case, location 2). As the CPU executes each instruction and retrieves or fetches the next-higher-numbered instruction from memory, the CPU "walks" its way through a program. The process of the CPU's "walking" through the program as it performs each task encoded in each instruction is called *program execution*.

Usually, the instruction number that the CPU will next execute is the next higher instruction. However, as a result of the outcome of the current instruction, the CPU can begin to execute instructions elsewhere in the series of numbered instructions. In effect, the CPU "jumps" from one series of instructions to another series. Perhaps the computed result of the most recent instruction marks a condition meaning that a sequence of instructions has successfully been completed. From the computed result, or from information encoded into

the instruction itself, the CPU sets its program counter to a new number of a new sequence and begins to execute that new sequence of instructions.

Cue:
A program consists of instructions that perform a general task.

A collection of instructions designed to perform a general task is a *program*. Although the program's task may range from word processing to playing chess, the CPU executes all programs' instructions in the same basic way. The CPU executes the program's binary instructions in a serial progression, and from time to time, the CPU "jumps" to a new point of serial progression.

You can think of this process of CPU program execution as you might picture the process of painting a house. The painter starts painting the house at some corner and paints from as far down and as far right as his arm can reach. (The CPU begins to execute instruction 1.) From the beginning point, the painter brushes as far up and as far left as his arm can reach. (The CPU executes a sequence of related instructions.) In a series of parallel horizontal brushstrokes, the painter progresses up the wall section. (In a series of instruction fetches, the CPU executes a logical block of a program.) After finishing the wall section, the painter moves to an unpainted area to begin a new section and repeats the process. (Likewise, the CPU completes a block of program instructions and jumps to a new block to begin a new function.) The presence of "conditions" such as windows or doors causes the painter to modify his painting sequence accordingly, but his progress would be considered orderly, not random. (The presence of IO conditions such as a key press or a disk-file requirement causes the CPU to alter the exact sequence of instructions being executed, but the CPU's progress is still orderly and based on the underlying logic of the whole program.)

The program COMMAND.COM is loaded into memory at boot time; the CPU executes COMMAND.COM's binary instructions. COMMAND.COM and external DOS command programs call on the services of the DOS *kernel*, which is a collection of program segments or routines. The kernel routines start at known memory locations. A DOS command (and other programs) uses special instructions that tell the CPU to interrupt the series of instructions of the command's program. The command's interrupting instruction also instructs the CPU to jump to the series of instructions of a kernel service routine.

The kernel service routine contains instructions that carry out a lower-level function such as writing a sector to a disk or reading a directory entry. The kernel uses interrupting instructions to jump BIOS service routines. BIOS routines are series of instructions that carry out very low-level tasks such as checking a disk drive for a ready status.

Through interrupts, higher layers of a working PC call on the next lower layer to perform tasks. As long as the calling method is uniform among layers, the higher layer does not have to know the instructions that the lower layer uses. The higher program layer can be generic to word processing, for instance; whereas the kernel layer can be more specific to operating system functions.

The BIOS layer can be very specific to the peripheral hardware. No layer knows the other layers' "business," but each layer relies on the layer below it to provide building-block services. The layers COMMAND.COM, the kernel, and BIOS have one common trait—their instructions' residence in the PC's memory.

Memory

Cue: *The values stored in memory are binary.*

The CPU fetches instructions from memory. Memory gets its name from its capability to retain values placed in individual memory locations. In all cases, the values that memory "remembers" are binary values, or patterns of 0's and 1's. For convenience and design efficiency, memory retains binary values in patterns of 8-bit (binary digit) or 1-byte locations. Each byte location is individually addressable by the CPU. Because each memory byte location has its own unique address, the CPU can access program instructions as a series of addresses. When the CPU develops values or data as a by-product of processing, the CPU can store in memory the values in addresses for access later.

Cue: *The CPU uses RAM for both reading and writing data.*

The CPU can store values in any location within the CPU's address range. To the CPU, the memory is randomly addressable. The CPU reads instructions from some randomly addressable memory locations and writes data to others. The name for memory that supports the CPU's reading and writing activity at random addresses is *random-access read-write memory*, or RAM. RAM is empty (has no stored values) when a computer starts. The previous work session's RAM content is gone. RAM loses its content when power is shut off in the system. Initially, RAM has no instructions or data to "remember." For the most part, RAM is loaded with program and data values from a disk drive. Some specialized RAM is powered by a backup battery that supplies power when the PC's main power is off. This type of RAM is called *nonvolatile RAM*, or NV RAM. The memory that retains the setup parameters of an AT-class computer is an example of NV RAM. Still, if NV RAM's battery power is turned off, NV RAM loses its content.

Cue: *The CPU cannot write to ROM.*

Not all memory within the address range of the CPU is RAM. Some memory has instructions and data permanently encoded into its byte contents. The permanent encoding ensures that this type of memory does not lose its content when the power is switched off. Permanently encoded memory is also randomly accessible by the CPU through address locations. Because the content of this type of memory is permanent, the CPU cannot use this memory to store values; the CPU can read the memory but cannot write to it. This permanent memory is termed *read-only memory*, or ROM. Most of the PC's basic input output system, or BIOS, is contained in ROM. ROM chips are called *firmware* because their contents are firmly established, though not absolute. Some ROM chips can be erased by ultraviolet light and reprogrammed with new values. This erasable type of programmable ROM is called EPROM. In a working PC, ROM and EPROM function in an equivalent fashion.

Buses and Control Chips

Cue:
Buses are electrical pathways through which the CPU signals its support circuits and memory.

Although the CPU supervises the activities of the PC, the CPU relies on a collection of support circuits. The CPU asserts its control by signaling support circuits over electrical pathways called *buses*. The specific bus through which the CPU coordinates computer activity is called the *control bus*. Memory locations are accessed through the *address bus*. The memory's contents are read and written through the *data bus*. Many components are connected to the buses in a party-line-like arrangement. Part of the function of the control components is to ensure that the correct device is the recipient or sender of internal information. With numerous components connected to the three buses, successful computing requires a high degree of internal coordination among components. Control components, through the supervision of the CPU, provide this coordination.

All the components in the system-unit hardware step in time to a single system clock. At regular intervals, the system clock sends pulses that synchronize the work cycles of the components. To work in harmony, all of the system hardware steps through its tasks by using the "beat" of the system clock as a timing cue. Computing events outside the synchronous environment of the system hardware, such as your pressing a key, have no way of "falling into step" with the smooth flow of clock-controlled activity. Outside events happen in their own time frames, not in the time frame of the system clock. Outside events are asynchronous to the system clock.

To service an outside event, the CPU must interrupt what it is doing and give immediate attention to the outside event. As soon as it can, the CPU executes a special interrupt service routine as a block of instructions. A special control component called the *interrupt controller* has the job of detecting external events and informing the CPU that the current series of instructions must be "put on hold." The interrupt controller then informs the CPU where the instructions that process the outside event can be found. The CPU is then able to begin executing the interrupt service routine to clear the interrupt condition. When the outside event is handled, the CPU picks up where it left off and continues processing. In effect, the interrupt controller has taken an asynchronous outside event and made possible the handling of that event by the synchronous system hardware. When more than one outside event needs the attention of the CPU, the interrupt controller has the capability to determine the order in which the outside events are to be handled by the CPU.

Programming Languages

Today, programmers rarely write programs at the CPU's binary language level, using 0's and 1's. If they need to write a program that directly represents machine code, today's programmers use an assembler program. The assembler program takes symbolic words, which represent the machine code but are

more like English, and produces machine code (patterns of 0's and 1's) as its output. This output is a binary program of machine code. The CPU can then follow these machine code instructions when executing the program. Even though a person programming in an assembly language uses English-like words rather than binary on and off (0 and 1) patterns to write a program, that person is still programming at a low level, in which one line of the source program represents a single machine instruction. Each line of code in the assembler source program does a machine-level step. The programmer has full control over the CPU, but programming at this low level requires the programmer to be extremely familiar with the characteristics of the CPU.

High-level languages (HLLs) are program-writing tools that remove most of the programmer's need to know much about the microprocessor. In fact, most HLLs like C, BASIC, COBOL, and FORTRAN have been adapted to many different microprocessors. As with assembly language programming, high-level language programmers write source programs in English-like text. The text describes in an exact way what the programmer intends the program to do. Unlike the assembler source program, the text instructions in the HLL program each result in more than one machine instruction. When the source program is completed, most HLLs compile the source program into binary instruction code. The microprocessor executes this code to do the programmer's intended work. Some HLL's source programs are not compiled but remain in their English-like state up to the moment that they are executed. A special HLL function called the *interpreter* then translates each line in turn into machine code for the microprocessor to execute. Interpreted HLLs eliminate the compile operation but require the special interpreter function.

Cue: *High-level languages enable you to write programs in English-like text and to specify more than one machine instruction per text instruction.*

Booting DOS

When you switch your PC on, causing it to beep in a few seconds and give you the DOS prompt, you have initiated a procedure known as *booting* your computer. The term *booting* is derived from the old concept that someone with no external help can "pull himself up by his bootstraps." In other words, someone can start with nothing and turn it into something useful. Needless to say, the act of creating something from nothing requires some wide interpretation of what "nothing" is. When you boot your PC, the PC is nothing on a scale of computing output. The potential is there, but you can't compute with it.

Cue: *PCs initiate a boot procedure when you turn them on.*

Much of this computing potential resides in three disk files. The file that contains the core of DOS, the file that contains the basic input and output additions for ROM BIOS, and the DOS command processor are all waiting on the disk to be loaded into the PC. Before the PC can load these files and make DOS available to you, the PC must test itself, initialize external hardware, and load the three DOS files from disk into memory.

A special ROM whose address starts at location 1 instructs the PC to access track 0 of the boot disk and read the boot sector into RAM. The boot sector contains a short assembly language program that finds and loads the two DOS system files. The boot program then looks for the CONFIG.SYS file in the root directory of the boot disk. If CONFIG.SYS is found, the boot program opens the file and installs into memory any device drivers that CONFIG.SYS refers to.

Through the loading of device drivers at the time of booting, the PC is given the provision to operate in a configuration tailored to the specific needs of the PC's individual hardware and running requirements. Whereas most device drivers are built into DOS's BIOS extensions, CONFIG.SYS device drivers are selectable and included under your control. CONFIG.SYS even has a provision to tell the boot program where to look for COMMAND.COM in the directory structure. The boot program locates COMMAND.COM in the CONFIG.SYS-designated disk location or in the boot disk's root directory. When COMMAND.COM is found, it is loaded into memory.

When COMMAND.COM is loaded into RAM, the boot program turns control over to COMMAND.COM by putting COMMAND.COM's first instruction location in the CPU's program counter. COMMAND.COM does not immediately display the system prompt and await your input, however. First, COMMAND.COM relocates part of its own instructions to another location in memory. In effect, COMMAND.COM splits itself into two parts. One part will remain in memory at all times. Another part, a transient part, will be overwritten by incoming programs that need the memory occupied by the transient part in order to fit into memory completely. The part of COMMAND.COM that stays in memory has the capability to reload the transient part from disk in order to be a whole program once again.

COMMAND.COM searches the root directory of the boot disk for a file named AUTOEXEC.BAT. If the AUTOEXEC.BAT file is found, its contents are executed as a series of DOS commands. By providing for this special batch file to be executed at boot time, DOS can tailor the start-up of every PC to meet the specific needs of the user. AUTOEXEC.BAT can contain any DOS command but normally contains commands such as PROMPT, PATH, TIME, and DATE. The PATH command is discussed in detail in Chapter 6, and the PROMPT, TIME, and DATE commands are discussed in Chapter 8.

When AUTOEXEC.BAT has been executed by COMMAND.COM, you will see the DOS prompt, or *system prompt* (assuming that one of the commands does not start a program). COMMAND.COM is then ready to receive your command.

Understanding Commands

The lines that you type and enter at the DOS prompt are centered on commands. Commands are not only DOS commands but also executable program file names. People don't think of executable program names as commands, because the word *command* in the context of DOS is more easily associated with internal and external DOS commands. COMMAND.COM expects to find some kind of command in every line you enter.

Reminder:
Commands can be DOS commands or executable program names.

Views of Command Names

In the most fundamental respect, commands are programs. Some commands are program blocks internal to COMMAND.COM. These programs are the so-called "internal commands." All other command programs are external to COMMAND.COM. Among the external command programs are the utility command programs of DOS. These external DOS commands are not built into COMMAND.COM but exist as disk files that COMMAND.COM loads and gives control to. The remaining external command programs are simply the applications programs that you use to do your computing.

Cue:
Commands can be internal or external.

Assuming that you have set a search path to a command's home directory, you simply enter the command's file name, and COMMAND.COM finds and loads the file. You don't enter a file extension as part of the command's name, because COMMAND.COM knows the allowable extensions for command files. COMMAND.COM recognizes as executable programs those files with extensions of COM and EXE.

You can't simply rename any file to a COM or EXE extension in order to make COMMAND.COM execute a file, though. COMMAND.COM will load such a renamed file, but the PC's operation will stop or hang. Only EXE and COM program files have the instructions necessary for execution.

COMMAND.COM has the capability to use the contents of a special text file as an input stream of commands. Text files with the BAT extension contain DOS commands that COMMAND.COM can execute. The AUTOEXEC.BAT file is such a text file. These executable text files are called *batch files*. Chapter 11 is dedicated to explaining batch file basics.

Cue:
A batch file is an executable text file.

COMMAND.COM does not turn control over to a batch file as it does an EXE or a COM file loaded in memory. Instead, COMMAND.COM remains in control as it interprets each line in the batch file and carries out the action associated with each line. From your point of view, you enter a batch file's root file name at the DOS prompt, and your computer does the file's intended work. The outward appearance of the executing batch file is not much different from the outward appearance of an executing COM or EXE program file. Because of this

similarity in appearance, you can think of a batch file as an executable command file. Just remember that COMMAND.COM steps through each line of a batch file as it executes individual command lines along the way, but that COMMAND.COM turns over control of the PC to a COM or EXE file.

If a command name is the root file name for three files in the same directory, and those files have the extensions BAT, COM, and EXE, COMMAND.COM selects the file with the COM extension as the file for execution. If a BAT and an EXE file are in the same directory and have the same root file name, COMMAND.COM executes the EXE file. You may want to keep this execution priority in mind when you name batch files. Naming a batch file with the same root name and in the same directory as an external DOS command requires you to include the desired file's extension on the command line. For convenience, you shouldn't duplicate a root file name between a BAT file and a COM or EXE file.

Now that you have learned that program names are technically command names, you can relax the technical definition. For the purpose of this book's discussion of commands, you can consider commands to be DOS commands, both internal and external. Applications programs will be called *applications programs* or simply *programs*. Batch files will be referred to as *batch files*.

Categories of DOS Commands

You can categorize DOS commands in more than one way—according, for example, to whether they are internal or external. Another useful way to categorize commands is by their function in the operating system. In the following sections, you will see commands categorized as internal versus external and again categorized by function.

The following is a list of MS DOS V4.01's internal commands; PC DOS has a very similar set of internal commands. Older versions of DOS have fewer internal commands. The internal commands do not contain individual program files. All the internal commands are contained in COMMAND.COM. COMMAND.COM knows how to access its internal block of program code, or routine, in order to execute an internal command.

BREAK	DEL (ERASE)	MKDIR (MD)	SET
CALL	DIR	PATH	SHIFT
CHCP	ECHO	PAUSE	TIME
CHDIR (CD)	EXIT	PROMPT	TYPE
CLS	FOR	REM	VER
COPY	GOTO	REN (RENAME)	VERIFY
CTTY	IF	RMDIR (RD)	VOL
DATE			

Because the internal commands are part of COMMAND.COM, they are available to you any time you see the DOS prompt. External commands must be available to COMMAND.COM from disk files. With the advantage of having internal commands built into COMMAND.COM, you may wonder why DOS doesn't include all commands within COMMAND.COM. The reason is memory conservation. COMMAND.COM occupies RAM in the same way that any program occupies RAM. For every byte that COMMAND.COM occupies, another program or function must sacrifice a byte. For the sake of memory conservation, many commands are external.

Reminder:
You can issue internal commands at the DOS prompt at any time; external commands are supplied to COMMAND.COM from disk files.

Because they are external to COMMAND.COM, external DOS commands don't have to be as conservative in memory use as internal commands. Although some external commands such as SHARE and MODE stay in memory after they are loaded by COMMAND.COM, the majority of the external commands are loaded, do their jobs, and exit completely. The following list shows the external commands that come with the generic version of DOS V4.01. As is the case with internal commands, the external commands included in different versions of DOS vary.

APPEND	DOSSHELL	JOIN	REPLACE
ASSIGN	EXE2BIN	KEYB	RESTORE
ATTRIB	FASTOPEN	LABEL	SELECT
BACKUP	FC	MEM	SHARE
CHKDSK	FDISK	MODE	SORT
COMMAND	FIND	MORE	SUBST
COMP	FORMAT	NLSFUNC	SYS
DISKCOMP	GRAFTABL	PRINT	TREE
DISKCOPY	GRAPHICS	RECOVER	XCOPY

Before COMMAND.COM can find and execute an external command or program, one of three conditions must be met:

1. The command must reside in the current directory of the logged disk drive.
2. You must have issued a PATH command containing as one of the search paths the external command's drive and directory.
3. You must provide an explicit path specifier in front of the command's name. The path specifier must include the drive and path information necessary to allow COMMAND.COM to locate the file on the file's disk.

On hard-disk-based systems, users prefer to give a PATH command to the external commands. A hard disk almost always has enough capacity beyond normal program and data files to accommodate the external DOS command files. With a search path to external commands established, most hard disk users don't see much difference between internal commands and external commands. Even hard disk users should have a bootable floppy with the external command files on it. You can never tell when a hard disk failure is going to occur.

For users of floppy-only systems, dealing with external commands is not so simple. Although you can specify a search path, you are likely to be swapping disks in and out of the drive(s), in which case the path setting doesn't know that the search disk is no longer in the specified drive. One solution in a two-floppy-drive system is to use drive A: for DOS and the external commands. Make drive A: the logged drive; the external commands will execute with just a name given to COMMAND.COM. Another solution is always to give the drive and, if necessary, the path parameter as a part of the command. You will soon get into the rhythm of giving an external command's name with the additional drive or directory preamble, and you'll have fewer false starts at the DOS prompt.

Cue:
DOS commands can be categorized according to function.

As you have become more familiar with DOS, you will have noticed that commands are related in categories of general function. When you have a general DOS function in mind, you can choose from a selection of commands that share common characteristics of that function. The functional categories of DOS commands are these:

Disk management *(see Chapter 5)*
Directory management *(see Chapter 6)*
File management *(see Chapter 7)*
System information management *(see Chapter 8)*
Input-output-device management *(see Chapter 9)*
Batch file management *(see Chapter 11)*
System configuration *(see Chapter 14)*
Country information management *(see Chapter 15)*

Part II of this book devotes a chapter to each of the first five functional categories. Parts III and IV cover the remaining three. Tables 3.1 through 3.8 summarize the commands included in each of the functional categories. Beside each command name is a brief description of the command's action. You will find these tables handy as a reference to commands from a function point of view.

Table 3.1
Disk Management Commands

Command	*Action*
CHKDSK	Analyzes disk directories and the disk's FAT. Reports disk space and memory status.
DISKCOMP	Compares two diskettes for differences.
DISKCOPY	Copies an entire diskette to a different diskette with the original diskette's format.
FDISK	Creates and maintains the partition table for DOS partitions on a hard disk.
FORMAT	Prepares a blank or recycled disk to receive files. Creates DOS's bookkeeping records on the disk.

LABEL	Creates, changes, or deletes a disk's volume label.
RECOVER	Recovers a single file or all of a disk's files when a file or directory has a bad sector.
REPLACE	Updates a disk with files of the same name from another disk.
SELECT	Installs DOS on a disk with country-specific information. Configures the DOS Shell in V4.
SYS	Installs the hidden DOS system files from one disk to another.
VOL	Reports a disk's volume label on the screen.

Table 3.2
Directory Management Commands

Command	*Action*
APPEND	Locates files needed by programs outside the current directory.
CHDIR (CD)	Makes a new directory current on a drive or displays a drive's current directory.
PATH	Establishes directory search alternatives for COMMAND.COM to use to load programs or external commands when the programs or commands are not located in the current directory.
MKDIR (MD)	Creates a new directory in a disk's directory tree structure.
RMDIR (RD)	Removes an empty directory from the directory tree structure.
TREE	Displays the directory tree structure of a disk.
XCOPY	Copies files in a selective manner. Includes the provision to create directories on the destination disk.

Table 3.3
File Management Commands

Command	*Action*
ATTRIB	Reports and modifies the archive and read attributes of a file or group of files.
BACKUP	Copies files in a special format from one disk to another to be restored when needed by RESTORE.

Table 3.3—*continued*

Command	*Action*
COMP	Compares two files or sets of two files and reports differences between files.
COPY	Copies a file or a group of files. Concatenates (joins) multiple files into one file.
DEL (ERASE)	Releases a file or group of files from a disk and returns the space occupied by the files to DOS.
FC	Compares two binary or text files and reports differences in selectable formats.
RENAME	Changes the root file name or extension of a file or group of files.
RESTORE	Reads specially formatted backup files produced in a BACKUP operation from a backup set disk to another disk.
TYPE	Sends the content of a file to the screen for viewing.
VERIFY	Turns DOS's internal read-after-write file verification routines on or off. Reports the current VERIFY setting.

Table 3.4
System Information Commands

Command	*Action*
CHCP	Displays and changes the current code page.
COMMAND	Starts a secondary command processor to allow running programs to accommodate DOS commands.
DATE	Sets or displays the system date.
DIR	Displays the directory information for files in a directory.
DOSSHELL	Starts the V4 graphical interface program.
EXIT	Returns from a secondary command processor to the one started below it.
MEM	Displays the current status of system memory.
MODE	Sets the operational parameters of many devices.
PROMPT	Sets the appearance of the system prompt.

SET	Establishes or displays the command processor's environment strings.
TIME	Sets or displays the system date.
VER	Reports the version number of the DOS program that booted the computer.

Table 3.5
Input-Output-Device Commands

Command	*Action*
ASSIGN	Directs disk requests for one disk to another disk.
CLS	Clears the screen and puts the cursor in the upper left corner.
CTTY	Provides for a serial device to be the standard input-output device.
FIND	Locates all lines in a file that contain a given parameter string. Displays the line(s) on the screen.
GRAPHICS	Enables some printers to print the content of an all-points addressable graphics display.
JOIN	Connects a drive to a directory on another drive to produce a single logical directory.
MORE	Filters screen output to produce a pause and a continuation prompt between screens of output.
PRINT	Outputs files to the printer device.
SORT	Filters output from a command and sorts the output alphabetically before relaying the sorted output to the standard output device.
SUBST	Allows use of an alias drive name in place of the drive's true name.

Table 3.6
Batch File Commands

Command	*Action*
CALL	Enables the nesting of batch files.
ECHO	Displays the command lines of an executing batch file.
FOR	Marks the repeating of DOS commands in a batch file.

Table 3.6—*continued*

Command	*Action*
GOTO	Causes batch file processing to continue on the line following a given label.
IF	Causes a line in a batch file to execute conditionally.
PAUSE	Temporarily pauses the execution of a batch file.
REM	Enables the addition of nonexecuting remarks in a batch file.
SHIFT	Enables the use of more than 10 replaceable parameters in a batch file.

Table 3.7
System Configuration Commands

Command	*Action*
BREAK	Controls the checking of the Ctrl-Break key press.
BUFFERS	Sets the number of system disk buffers allocated.
DEVICE	Names a device driver to be used at boot time.
FCBS	Specifies the number of V1. file control blocks that are allowed to be open at once.
FILES	Specifies the number of files that can be open at one time.
FASTOPEN	Enables the internal FAT buffering feature and the directory read-ahead feature.
INSTALL	Enables CONFIG.SYS processing to load a command.
LASTDRIVE	Sets the maximum number of drives that can be accessed.
SHELL	Sets the name and location of the command processor that DOS uses during a boot in place of COMMAND.COM.
STACKS	Enables the overriding of the default stack allocation.
SWITCHES	Makes an extended keyboard appear as a standard keyboard to DOS.
SHARE	Loads and enables disk-change checking.

Table 3.8
Country Information Commands

Command	*Action*
CHCP	Selects character-set code pages for IO devices.
COUNTRY	Modifies country-dependent settings.
DISPLAY.SYS	Supports code-page switching on certain displays.
GRAFTABL	Accommodates national language characters displayed in graphics mode.
KEYB	Selects a special keyboard key layout.
NLSFUNC	Supports country-specific information.
PRINTER.SYS	Supports code-page switching on certain printers.

Understanding the Command Line

COMMAND.COM is the DOS command interpreter. All DOS commands that you issue go through the command interpreter to be executed. The phrasing, or *syntax*, of each command line is important to COMMAND.COM: the syntax of the command line is the input that COMMAND.COM interprets. The following sections discuss the issuance of commands and provide information relating to all commands. This discussion is fundamental to later chapters. If you are already comfortable with the information presented here, read the following sections for a quick review. If you find the information new, be sure that you understand it before you move on.

Cue:
The syntax of a command line is the way in which the command line is phrased.

Understanding Parameters

Most DOS commands require that you provide additional text on the command line. The text you supply to indicate the objects of a command's actions is called *parameters*. COMMAND.COM reads your command line and breaks the line's text apart into the line's meaningful parameters. This process of breaking the line apart is called *parsing*. COMMAND.COM parses the command line by using the rules of DOS syntax that the command dictates. Each DOS command has its own rules of syntax. By parsing the command line, COMMAND.COM determines what you want to do. COMMAND.COM parses the line's command and then the command's parameters.

Cue:
Parameters specify the objects of a command's actions.

Some commands have a single parameter. The VERIFY command, which controls global file-writing verification, has one optional parameter, which indicates the state of VERIFY. The parameter is either ON or OFF. COMMAND.COM

parses a VERIFY command line by looking for an ON or OFF value as the single parameter. If no parameter is found, DOS displays VERIFY's status.

File-copying commands such as XCOPY, REPLACE, and COPY each require two parameters. All three of these commands have a source parameter and a destination, or *target*, parameter each. The DIR directory listing command has one parameter: the source directory to list. Think of the COPY command, for example, as meaning the following:

COPY *from the source parameter to the destination parameter.*

Think of the DIR command as meaning this:

Display a DIR*ectory listing of the source parameter.*

In an actual command line, the COPY command may look like the following:

```
COPY C:\MEMO\MYFILE.123 A:\MYFILE.BAK
```

The DIR command may look like this:

```
DIR C:\DOS4\MYJOB.BAT
```

Cue:
A delimiter separates portions of the command line; a specifier is part of a parameter.

You can see that the command lines for both COPY and DIR have the command names as the first "word." A space separates the command name from the rest of the line. To COMMAND.COM, the space marks the end of a word or phrase on the command line. The space therefore acts as a *delimiter*. Notice in the COPY example that a space also delimits C:\MEMO\MYFILE.123 and A:\MYFILE.BAK. The second space delimits the command's source and destination parameters. Within each parameter you notice punctuation characters such as the colon (:), backslash (\), and period (.). COMMAND.COM expects to find the component parts of each parameter delimited with certain punctuation characters such as these. The component parts of parameters are called *specifiers*.

DOS uses the specifiers within parameters to extract the details of the parameters. Each parameter in the COPY and DIR examples contains three specifiers. The A: and the C: are disk drive specifiers that tell DOS where the source or destination parameter is located. Remember that DOS is a multidisk operating system. The drive name specifier in the command's parameters gives you control of which disk DOS will use.

The next specifier in the sample COPY and DIR commands is the path specifier, indicated by the \ character. The \ delimits each directory level in a full path name for a file. When DOS sees \MEMO\ in the sample COPY command, DOS interprets the \ characters as meaning that the word MEMO is part of (and possibly all of) a directory name.

The final specifier is the file specifier. MYFILE.123 is the source file specifier in the sample COPY command. MYFILE.BAK is the destination file specifier.

Within the file specifier, COMMAND.COM can separate the root file name from the extension by finding the period (.) before the extension.

As you can see, COMMAND.COM uses rules to parse commands that contain file parameters as well as to parse commands that contain other types of parameters. When you issue DOS commands, you must form the command line properly in order for COMMAND.COM to parse the parameters of the command line correctly. In addition to parameters, some commands include optional modifiers of the command's basic action. These modifiers are called *switches*.

Cue: *Switches are optional modifiers of the command's action.*

Understanding Switches

Not every command has switches, but many commands do. DOS commands all have basic actions that make the commands useful. Switches add additional selectivity or action to the basic action of a command, making the command more versatile and therefore even more useful. Switches usually follow the parameters on a command line and are delimited by a leading / character. COMMAND.COM parses the text that follows the / character as the switch contents. In most cases, only one character follows the switch's / delimiter. Some commands include switches that may have several characters following the /. In all cases, the text that accompanies the / character in the command line has no space characters within the switch text. DOS would interpret an intervening space as a delimiting character and thus misread the switch.

The DIR command has two switches. The DIR /W switch displays a directory listing in a *wide* format. The DIR /P switch displays a directory listing one screen *page* at a time and prompts you before displaying another page. You can issue the DIR command with either or both switches. Most commands with multiple-switch capability can be issued with more than one switch appearing in the command line. Some commands' switches are mutually exclusive, however, which means that they can't be used together in the same command line. DOS objects with an error message when certain command's switches are issued together. Other commands have switches that make sense only when paired with other switches. The BACKUP switch /T for *time* makes sense when used with the /D *date* switch. The BACKUP /T switch alone doesn't produce any meaningful modification of the BACKUP command, because basing a backup on the time of day that a file was created or modified is rather random. Even though the switch doesn't make sense, DOS will not object with an error or warning message.

From command to command, the same letters may be used for switches. Most often, the same literal switch letter from one command will carry a different meaning when issued in another command; commands that use switches use their own set of switches. When you use switches, make sure that you understand the meaning of each switch as it applies to each command.

Understanding Defaults

Cue:
DOS uses a default value if you have the opportunity to specify a value but don't supply that value.

A *default* is a value that DOS uses if you don't specify a value. Command-line parameter specifiers are often optional. Not surprisingly, you don't have to include optional specifiers in a command line, but you can include them if you want. COMMAND.COM will not object. The rules of syntax for each command determine which parameters and specifiers are required or mandatory. Those parameters and specifiers that are not required are optional.

When a disk drive specifier is optional, DOS uses the current disk drive as the default value for the drive specifier. When you boot your PC, the boot disk drive is the default disk drive. DOS logs you to the boot disk drive. The logged disk drive is your current disk drive. The *logged disk*, the *default disk*, and the *current disk* all refer to the same thing. You change the current disk drive to another disk drive by typing the desired drive's letter, followed by a colon, and pressing Enter. At the command prompt, the process looks like this:

```
A>C:

C>
```

You can have only one default (current or logged) drive at any time. DOS uses the current drive as the default drive when you omit the optional drive specifier from a command line.

Each disk has its own file system. When you boot your PC, DOS logs to the root directory of each disk on your system. The root directory is the current directory of each disk drive at boot time. Assuming that you have created subdirectories on each of your system's disks, you can change each disk's default directory to a different directory. Changing directories is similar to logging to a new drive except that only one drive can be DOS's default drive. Each drive can be logged to its own default or current directory. The logged drive's current directory is your working directory. If you omit the optional drive specifier and the optional path specifier from a command, DOS uses the current working directory. If you include a drive specifier but omit the optional path specifier, DOS uses the specified drive's current directory.

Defaults help to make command lines look simpler. If you issue only the DIR command on the command line, COMMAND.COM still parses the line for specifiers. In the DIR command, the drive, the path, and the file name specifiers are all optional. DOS supplies the current default values when you don't issue those optional specifiers. If you mainly use a hard disk, you may seldom use a drive specifier in a command line; if you mainly work in a particular subdirectory, you may seldom use a path specifier in a command line. In such a case, DOS supplies default values for the command line. COMMAND.COM still parses the command line according to the rules of syntax, however. If COMMAND.COM finds that you have specified an optional specifier, the specifier that you supplied is the one used.

Using the Keyboard

Cue:
The keyboard is the standard input device; the screen is the standard output device.

Normally, you issue DOS commands from the keyboard and view the commands' messages on your screen. COMMAND.COM gets command lines from the standard input device: the keyboard. COMMAND.COM issues system messages through the standard output device: the screen. Error messages produced by DOS during command execution also are output through the standard output device. However, DOS provides for input or output to be associated with a device other than the standard device. You can instruct DOS to redirect the standard inputs and outputs to other logical DOS devices. Chapter 4 discusses the process of redirection. For now, consider the keyboard the input device for DOS commands.

Just like a typewriter, the keyboard of a computer contains all the letters of the alphabet. The numbers, symbols, and punctuation characters are virtually the same. The computer keyboard has the familiar QWERTY layout. (The name QWERTY comes from the letters found on the left side of the top row of letters on a standard typewriter.) But a computer keyboard is different from a typewriter keyboard in several important ways.

The most notable difference is the "extra" keys—the keys that do not appear on a typewriter. These keys are described in table 3.9. Depending on the type of computer you use, you will also find 10 or 12 special function keys.

Table 3.9
Special Keys on the Computer Keyboard

Key	*Function*
Enter	Signals the computer to respond to the commands you type. Also functions as a carriage return in programs that simulate the operations of a typewriter.
Cursor keys	Changes your location on the screen. Included are the arrow, PgUp, PgDn, Home, and End keys.
Backspace	Moves the cursor backward one space at a time, deleting any character in that space.
Del	Deletes, or erases, any character at the location of the cursor.
Ins	Inserts any character at the location of the cursor.
Shift	Enables you to capitalize letters when Shift is held down and the letter is typed. When pressed in combination with another key, can change the standard function of that key.

Table 3.9—*continued*

Key	*Function*
Caps Lock	When pressed to the lock position, capitalizes all characters to be typed. Caps Lock does not shift the keys, however. To release, press the key again.
Ctrl	The Control key. When pressed in combination with another key, changes the standard function of that key.
Alt	The Alternate key. When pressed in combination with another key, changes the standard function of that key.
Esc	In some situations, enables you to "escape" from a current operation to a previous one. Sometimes Esc has no effect on current operation.
Num Lock	Changes the numeric pad from cursor-movement to numeric-function mode.
PrtSc	Used with the Shift key to send the characters on the display to the printer.
Print Screen	Found on Enhanced Keyboards. Same as Shift-PrtSc.
Scroll Lock	Locks the scrolling function to the cursor-control keys. Instead of the cursor's moving, the screen scrolls.
Pause	Suspends display output until another key is pressed. (Not provided with standard keyboards.)
Break	Stops a program in progress from running.
Numeric keypad	A cluster of keys to the right of the standard keypad keyboard. The keypad includes numbered keys from 0 to 9 as well as cursor-control keys and other special keys.

Many of these extra keys are designed for use in combination with other keys. For example, pressing the Shift key and the PrtSc key in combination causes DOS to print the contents of the current screen. Pressing the Ctrl and PrtSc keys simultaneously causes DOS to continuously print what you type and what DOS sends to the screen. On some keyboards, pressing only the Print Screen key causes the contents of the screen to be printed. Pressing Ctrl and PrtSc a second time turns off the printing.

The function keys are shortcuts. Not all programs use these keys, and some use only a few of them. When used, however, they automatically carry out repetitious operations for you. For example, the F1 key is often used for on-line help in application programs. On-line help displays instructions from the computer's memory to help you understand a particular operation. The DOS V4.0 shell uses the F3 key to automatically back out of one operation and move into another. The F10 key moves the cursor to various parts of the screen in the DOS Shell.

Each character you type is converted by the keyboard electronics into a code that the computer can process. A key's code—not the key's position or the legend on the key itself—is significant to DOS.

Many early PC-compatible computers use a standard keyboard like that of the IBM PC. Other machines use Personal Computer AT keyboards. IBM's PS/2 computers use the 101-key Enhanced Keyboard. You can determine whether your computer has a standard keyboard, a Personal Computer AT-style keyboard, or an Enhanced Keyboard. Certain keys are found only on specific keyboards. For example, you find the Print Screen and Pause keys only on the Enhanced Keyboard. You can, however, simulate these keys by using a combination of keys on the standard keyboard. Remember that a key's code—not its position or legend—is important to DOS.

Some new keyboards enable you to change key caps and to switch key definitions for the Caps Lock, Ctrl, Esc, and (~) keys. Northgate Computers, for example, not only offers these options but also offers an enhanced keyboard. On this enhanced keyboard, the first 10 function keys are to the left of, instead of across the top of, the alphabet and number keys. The arrangement requires one more key than do the 101-key enhanced keyboards. Thus, the 102-key keyboard.

Small computers, such as "lunchboxes" and laptops, may use nonstandard keyboards, usually to conserve space. On some, space is so restricted that you need an external numeric keypad for use with software that performs advanced calculations.

DOS acts on combination keys in predefined ways. Table 3.10 lists key combinations and the action that DOS takes when you use a key combination.

Table 3.10
DOS Key Combinations

Keys	*Function*
Ctrl-Num Lock	Freezes the display; pressing Ctrl-S or any other key restarts the display.
Shift-PrtSc or Print Screen	Prints the contents of the video display (print-screen feature).
Ctrl-PrtSc	Sends lines to both the screen and the printer; giving this sequence a second time turns this function off.
Ctrl-C or Ctr-Break	Stops the execution of a program.
Ctrl-Alt-Del	Restarts DOS (warm boot)

DOS uses some of the function and editing keys for editing a command line. When commands are repetitive, you can use the editing provisions of the assigned function keys to make the task of issuing commands easier. Table 3.11 lists the function keys and the action the keys take at the command line prompt.

Table 3.11
DOS Command Line Editing Keys

Key	*Action*
Tab	Moves cursor to the next tab stop
Esc	Cancels the current line and does not change the buffer
Ins	Enables you to insert characters in the line
Del or Backspace	Deletes a character from the line
F1 or right arrow	Copies one character from the preceding command line
F2	Copies all characters from the preceding command line up to the next character you type
F3	Copies all remaining characters from the preceding command line
F4	Deletes all characters from the preceding command line up to, but not including, the next character typed (opposite of F2)
F5	Moves the current line into the buffer but does not allow MS-DOS to execute the line
F6	Produces an end-of-file marker (^Z) when you copy from the console to a disk file

DOS converts all lowercase command line characters into uppercase characters. You can therefore use upper- or lowercase when you are working in DOS. The FIND command is an exception in that the FIND command's string specifier is interpreted as being supplied from the command line. FIND attempts to match the string specifier with text in a file. To accommodate matching lowercase text, FIND accepts a lowercase search specifier. FIND is a special command that DOS calls a filter. FIND and other filter commands are discussed in Chapter 9.

Stopping a Command

DOS monitors the status of the Ctrl-C and Ctrl-Break key combinations to see whether one of the combinations has been pressed. If DOS detects that one of these combinations has been pressed, DOS stops the process that is is executing. If the process is the acceptance of a command from the keyboard, COMMAND.COM flushes the current line and issues a fresh prompt on the next line. If the process is the execution of a command, DOS stops the command at the first opportunity and returns the command prompt.

Reminder:
Pressing Ctrl-C or Ctrl-Break stops a command.

Some commands can be effectively stopped. For instance, a long directory listing resulting from a DIR command can produce several screens of output. If you press Ctrl-C during the listing, the listing operation will stop, and COMMAND.COM will prompt for the next command.

Other commands can be stopped, but not before they do some damage. For instance, if you mistakenly issue a FORMAT command with the wrong drive specifier, you can stop FORMAT. However, there is a good chance that FORMAT operated long enough to overwrite some of the DOS tables on the target disk. The disk will not be usable, and the disk's previous content is lost. The moral here is to issue commands carefully. Ctrl-C or Ctrl-Break will most likely be unable to stop a DOS operation quickly enough to keep some commands from overwriting DOS tables or user files.

Understanding the Environment

You interact with COMMAND.COM by issuing commands. COMMAND.COM carries out other duties in the working PC, such as loading and executing programs. Often only one program resides in memory at one time. Because DOS is a single-tasking operating system, only one program can run at one time. COMMAND.COM, as well as other specialized programs, can load additional programs into memory, however. In fact, some external DOS commands are programs that reside in memory along with applications programs or COMMAND.COM.

Cue:
The environment is a DOS storage area for values held in named variables.

When every program is loaded into memory, the program is given a set of values that DOS stores in reserved memory locations. The name for the storage area is the *environment*. COMMAND.COM establishes an environment to hold values assigned to named variables. Some DOS commands such as PATH and PROMPT place values in the environment when you issue the command. Other values are assigned by you when you use the SET command. Consider this sample SET command:

```
SET FINDIT=C:\DOS
```

The variable named FINDIT is assigned the character value of \DOS. The variable FINDIT and the variable's value C:\DOS are then part of the environment. Collectively, all the variables and their values are called the environment *string*. You don't just arbitrarily put variables in the environment. You put values in the environment so that they will be available to any program operating on the PC. The word *environment* has the connotation "surroundings." To a program and some DOS commands, the values that are part of the "surroundings" tell the programs how to perform certain aspects of the programs' jobs.

A database program, for instance, may need to know in which directory the data files are located. The program can search the environment string, looking for the predetermined variable FINDIT. On locating FINDIT, the program can extract FINDIT's value. The program can then use the variable's value as the directory to search. Needless to say, programs must know to use the environment, and you must know how to assign values to variables that programs expect. Your applications program's instructions tell you whether the program can use the environment's capabilities.

Because some DOS commands use environment variables, you will see references to the environment throughout this book. Chapter 8 has specific references to the environment in discussions of the SET and COMMAND commands. For now, just be aware that one of COMMAND.COM's important "behind the scenes" jobs is to make the environment available to the programs and commands that COMMAND.COM starts.

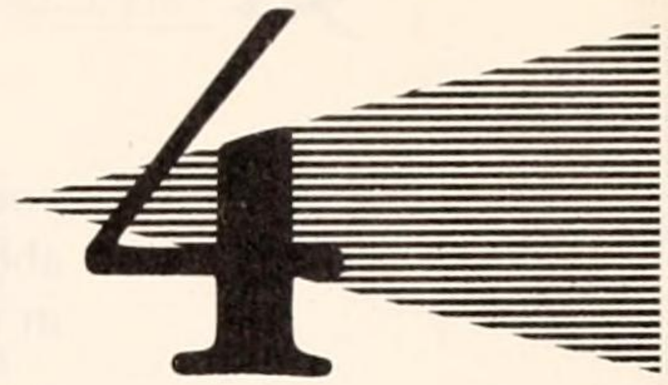
Summary

This chapter has discussed COMMAND.COM, the DOS command processor. The information presented in this chapter will help you as you work with commands. When you read about the core DOS commands in Part II and about other commands in Parts III and IV, you will be better equipped to use COMMAND.COM's universal concepts with those commands. In this chapter you learned the following:

- The CPU develops data as it executes instructions fetched from RAM and ROM. The data is stored in RAM.
- The CPU can interrupt its current instruction sequence and execute other logical blocks of instructions as service routines.
- Programs and service routines are written in computer languages. The lowest-level language is machine language that the CPU directly understands.
- The process of booting DOS is a series of operations during which the PC initializes and tests the hardware, loads the DOS files, and executes COMMAND.COM.

- Commands can be categorized according to whether they are internal or external and according to general DOS management function.
- COMMAND.COM parses a command line in order to extract the command, the parameters, the specifiers within parameters, and the switches.
- Many commands use default values for specifiers if you omit the specifiers in the command line.
- DOS commands use the keyboard for standard input and the screen for standard output.
- Besides the "typewriter" keys, a PC keyboard has keys and key combinations that DOS gives special meaning to.
- COMMAND.COM passes the values of environment variables to programs when COMMAND.COM loads the programs.

In the next chapter, you will learn how DOS manages the devices that accompany the personal computer.



Gaining Control of Hardware

Chapter 1 discussed the many contexts of the term DOS and the layered computer. In Chapter 2 you saw how DOS, through the kernel, implements a multidisk hierarchical file system. Chapter 3 discussed COMMAND.COM as the command processor layer of DOS. Chapter 3 also introduced the way the CPU steps through instructions for entire programs or for service routine blocks. This chapter provides the basis for understanding how DOS deals with devices. In DOS, a discussion of devices can include topics ranging from how COMMAND.COM accommodates redirection to how CONFIG.SYS can expand the device base of the PC to what kind of additional peripheral hardware and memory expansion your PC will support. Understanding devices not only returns DOS benefits but also helps you understand personal computing in the context of networking, windowing environments, and future operating systems.

Reviewing the Layered PC

In Chapter 1 you read about the layers of a working PC. When you sit down at the keyboard and run your programs, you are interacting with a higher layer of the PC. In this higher layer, you see computing as spreadsheet cells, word processor function keys, and modem links to on-line data services. This surface activity between you and your keyboard and screen is quite an abstraction from the logical and electromechanical activity going on at lower layers of the PC. Your primary interest is with the activity at hand. When you hit a golf ball or return a tennis ball, you don't often stop to study the underlying physics involved at the moment of impact. Your interest is placing the ball close to (or in) the hole or getting the ball over the net to a strategic area of the court. In computing activity, you are interested in creating an accurate financial model of next year's sales or reformatting a block of text for a manual. Just the fact that

you can use a personal computer as a tool to do useful work without worrying about how the computer operates is a tribute to how far computers have come in the past decade. The internal processes of the machine are hidden from view.

When you issue DOS commands at the DOS prompt to manage your disks, directories, files, and devices, you are at a slightly lower and slightly less abstract level of the computer. While working with COMMAND.COM, you are more aware of the file system and the devices on your PC. At the DOS prompt, you maintain the PC so that it will be more orderly and useful for your computing work. Just as you maintain your tennis racket or your golf clubs, you maintain your files and your PC configuration. In both cases, your activity accommodates the primary reason you have the equipment: the effective use of the equipment.

Cue:
The kernel translates higher-layer requests into requests for lower layers.

DOS maintains organizational order in your PC below the visible surface of COMMAND.COM. This core layer of operating system organization is the DOS kernel, which is less abstract than COMMAND.COM. The kernel deals with requests from COMMAND.COM and programs for data from files and devices. From file and device names that you or the program provides, the kernel translates these requests to logical sectors, starting clusters, and RAM buffer locations. The kernel translates input or output requests from the higher layers into uniform requests of a predetermined form for lower layers. The higher layers do not know the details of how the kernel handles the higher-layer requests. The kernel, in turn, does not know how the lower layers go about fulfilling the kernel's request. Yet thanks to the kernel, a program can be totally independent of the overhead of managing the file system and can count on its requests, such as for file input or printer output, to be fulfilled in a constant and reliable fashion.

Cue:
The BIOS is responsible for getting input from and output to the peripheral hardware.

Below the kernel layer, a layer of basic input-output, or BIOS, services is built on the hardware itself. The BIOS layer is the PC's device-dependent layer. The responsibility for getting input from and output to the peripheral hardware rests on the BIOS. Since the BIOS processes kernel requests through uniform service routines, the kernel is device-independent. Any new peripheral hardware that the BIOS supports is available for kernel requests. The BIOS, the least abstract layer, deals with peripheral hardware at the binary level. BIOS routines are the glue that attaches hardware to the basic computer in a fashion that disguises the seams.

Understanding Device Drivers

Cue:
Device drivers enable you to attach peripheral hardware to the DOS kernel.

Device drivers are specialized program modules or routines that accommodate the attachment of peripheral hardware to the DOS kernel. Most peripheral hardware such as disk drives, serial printers or modems, parallel printers, video displays, and mice connect to the PC's bus system through adapters or controllers.

Figure 4.1 shows how a typical program's I/O request is fulfilled by a device driver. The program, represented by the top block, makes a request of the DOS kernel. The kernel consults a list of device drivers to determine which should handle the request. The kernel views the device drivers as logical devices and transfers the program's request down to the device driver. The device driver, in turn, translates the kernel's logical request into a set of control and transfer actions by dealing directly with the device. The device's output data is transferred to the program through the device driver and the kernel. No one block knows exactly how the block below it works. Each block relies on a uniform interface to adjacent blocks to complete the I/O service.

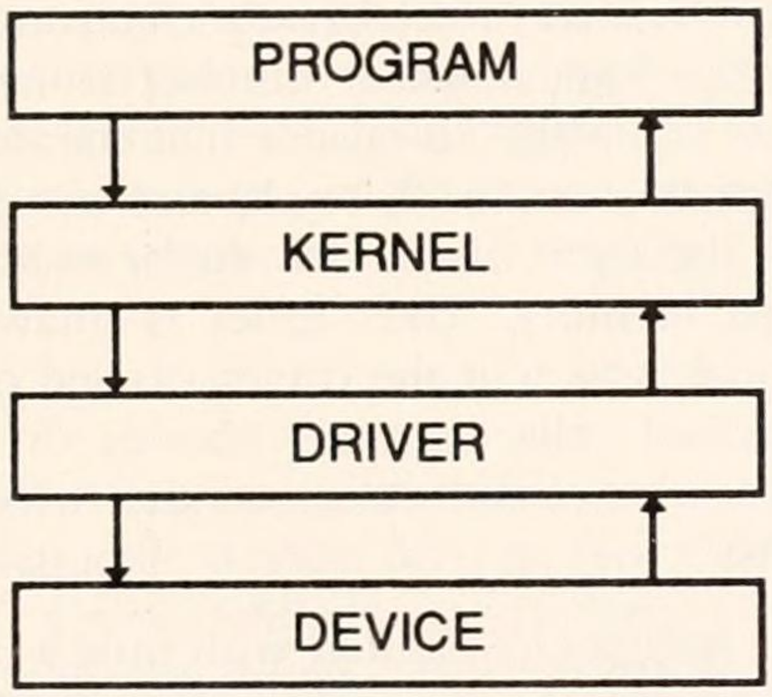


Fig. 4.1

How program I/O requests are filled.

Understanding Controllers

Cue:
Adapters, or controllers, control a physical device.

Adapters are electronic circuit boards that contain the necessary support logic to control a physical device. Adapters are often called *controllers* because the circuit's function is that of primary device control. Device drivers are written to manage the activities of the controllers.

Display adapters are examples of controllers. A PC may have one or more display adapters such as a Monochrome Display Adapter (MDA), Color Graphics Adapter (CGA), Enhanced Graphics Adapter (EGA), or Video Graphics Array (VGA). Each display adapter has its own supporting device driver. When any of these adapters are in text mode, they can display characters, numbers, and symbols all based in ROM. The characters, numbers, and symbols are encoded into ROM as patterns of bits that correspond to points of light, or pixels, on the display screen. Each character is composed of a rectangular grid of possible

pixels, with the pixels that form the character (or number or symbol) illuminated. The number of possible pixels in each character grid varies among adapters. The driver for each adapter enables the DOS kernel (and therefore a program) to request a character for placement on the screen. When you press a key and see the character echoed on the screen, you are seeing the work of the driver and the adapter. If you press the A key, you see an A on the screen even though the pixel density of the A is not the same as that of another display type. The drivers for the two types of display adapters shield the kernel from the details of how the A is produced on the screen.

Disk controllers are another example of adapters. The disk drive itself has a limited electronic function. The disk controller supplies much of the underlying logic or intelligence of disk operations. Although disk drives are fundamentally alike, each type of drive has its own electromechanical personality. The disk driver routines are adapted to the drive's electromechanical personality. When the kernel requests a logical sector number from the disk device driver, the driver manages the controller to ensure that the drive is ready for transfer and has been given the correct head, track, and sector number. When the drive controller returns the data, the device driver supervises the placement of the returned data into memory. The kernel is unaware of the nature and the number of individual tasks that the driver carried out to make the sector available. When the kernel calls on a disk device driver, the kernel can make a generic request for sectors and count on the driver to be responsible for the variation of the disk.

If a programmer is sufficiently familiar with how a new item of peripheral hardware works, the programmer can write a device driver for the hardware (or its adapter), and the PC can incorporate the new device. DOS clearly defines the programmatic interface between the kernel and the device driver. If a programmer strictly follows the definition, DOS can use the device.

Grouping Device Drivers

Physical devices as diverse as disk drives and keyboards tax the definition of a single type of device driver interface to the kernel. DOS provides for device drivers to fall into one of two groups: *character* device drivers are designed for devices such as the keyboard and printer; *block* devices are designed for devices such as disk drives and tape drives.

Character Device Drivers

Cue:
Character devices, such as the keyboard and display, perform I/O one character at a time.

Character devices perform I/O one character or byte at a time. Typical character devices in PCs are the keyboard, the display, serial ports, and printer ports. Normally, one device driver controls one character device. The device driver for a character device supplies a one- to eight-character device name to the

DOS kernel. The character device name is quite similar to a root file name in that both can have as many as eight characters in the root name. Character device names do not include extensions, however. Programs that use character devices treat the devices as though the devices were files.

The DOS kernel transfers characters to or from the character device driver by using RAM to store or buffer the characters temporarily. In some cases, the kernel checks the character stream for a ^C (Ctrl-Break) or a return (Enter) character. In other cases, the kernel passes a stream of serial characters to or from the character device driver and does not look at the actual characters. The kernel can also echo to the screen each character received from the character device driver. The actual behavior of the kernel-to-device-driver interface during character I/O is determined by the program initiating the requests for I/O.

Character device drivers support the less complex peripheral hardware. Although the character-oriented peripheral may not be complex in a computer context, character-oriented devices are too numerous and difficult to supervise for the DOS kernel to see to their detailed requirements. Yet the more complex device drivers that move massive amounts of data to and from devices such as disk drives are not well served by character device drivers.

Block Device Drivers

Cue: *Block device drivers transfer data to and from the kernel in large units.*

To deal with the movement of massive amounts of I/O data, DOS incorporates block device drivers. Like character device drivers, block device drivers are program routines designed to a well-defined programmatic convention. Unlike character device drivers, block device drivers transfer data to and from the kernel in large units of data called *blocks*; hence the term *block device driver*.

Block device drivers interface peripheral hardware to the kernel as blocks of addressable data. The most common type of block device driver is the disk device driver. On the disk, the addressable blocks of data are the data sectors. Block devices are given names that are letters, such as A, B, and C. These letter names are the same kind of names that DOS gives to your disk drives.

Unlike character devices, which DOS treats like logical files, block devices are considered logical disks by DOS. In fact, through the magic of the device driver, one physical disk drive can be logically addressed as several logical disk devices. Even a nondisk emulation of a disk's function can be incorporated into a block device driver. The VDISK device driver, for instance, uses a portion of system RAM that appears to the DOS kernel as a disk drive complete with sectors. Thanks to the special VDISK block device driver, a program can access a file from the virtual disk without any awareness that the requested file is actually a block of RAM.

Character and block device drivers form the backbone of the PC's input-output system. By hiding the hardware detail from the kernel yet interfacing with the kernel in a uniform fashion, device drivers make the hardware modular. Modularity is why the same program can run on an IBM PS/2 Model 80 or a Dell 310. Uniformity of driver design is why an optical-character-recognition device can transfer characters to a program through a serial port. In an hour, the same serial port may be receiving a download file through a modem.

Defining Resident and Installable Device Drivers

Cue:
Device drivers are either resident or installable.

The DOS system disks contain files that collectively make up the device drivers for a PC. Not all device drivers are supplied with the DOS package. DOS can't possibly anticipate all of the hardware that you have or will have. Additional device drivers can be added at any time, however, to the ones that come with DOS. On the DOS disk, the hidden system file IO.SYS (IBMBIO.COM in PC DOS) contains the device drivers that always load into memory at boot time. Because IO.SYS is part of the DOS-supplied software, the device drivers included in IO.SYS are called the *resident drivers*. In V2.0 and later, the DOS package contains other files that are device drivers. These device driver files have SYS extensions and include ANSI.SYS, VDISK.SYS, PRINTER.SYS, and others. Not every version of DOS has all these driver files. The SYS files are loaded into memory as part of the boot processing of the CONFIG.SYS file. The controlled way that the drivers are loaded into memory is called *driver installation*. The device drivers with the SYS extensions are therefore called *installable device drivers*. In a DOS environment, device drivers are either resident or installable.

Resident Device Drivers

Cue:
Resident device drivers reside in IO.SYS.

Resident device drivers reside in the hidden system file IO.SYS (IBMBIO.COM). Each version of DOS has its own selection of resident device drivers. Different implementations of DOS from PC vendors may modify the list of IO.SYS-included device drivers to include a particular hardware feature. By and large, though, each version of DOS has a distinct set of device drivers in IO.SYS.

Figure 4.2 depicts the resident device drivers available to the DOS kernel (and therefore the program) as a standard set of devices below the kernel. These devices drivers are viewed by the kernel as being logical devices capable of providing input, supplying output, or both.

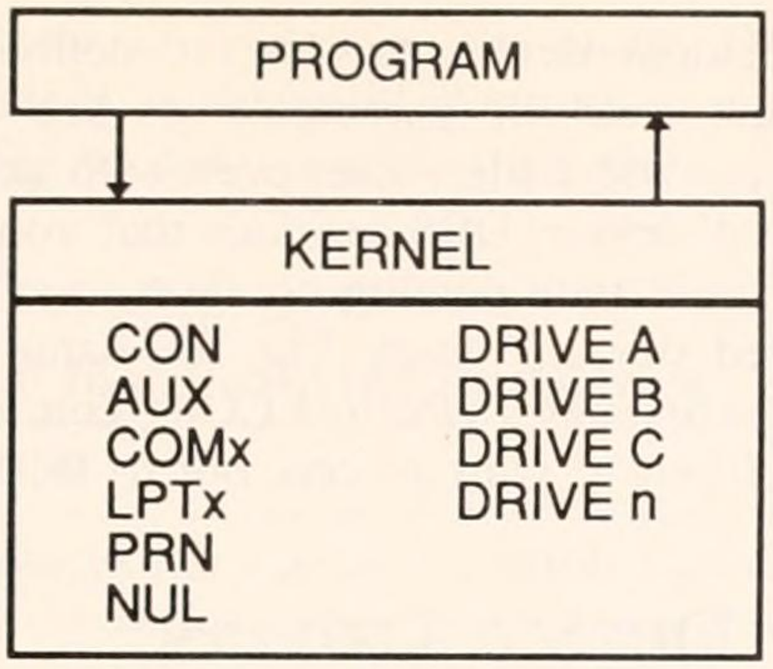


Fig. 4.2

The resident device drivers available to the DOS kernel.

The disk block-device-driver routines support the standard disk devices of the PC. As newer, higher-capacity floppy drives became standard equipment, new versions of DOS-included drivers support these disks. If you are running V3.1 and want to install a 720K 3 1/2-inch disk drive, for instance, DOS will not support the added drive. Your PC is most likely not the problem; the older version of DOS is the problem. As newer devices are added to PCs, the resident drivers are modified to keep pace.

DOS provides many resident character device drivers. As is the rule with character device drivers, the resident character device drivers reserve names for the devices they control. Table 4.1 lists the resident character device drivers for PC DOS V4.0. Other versions and implementations have most of these resident device drivers.

Table 4.1
PC DOS V4.0 Resident Character Device Drivers

Device Name	*Meaning*
CLOCK$	The system clock device driver.
CON	The keyboard as input and the display as output.
AUX OR COM1	The first serial communications port.
COM*n*	The *n*th serial communications port up to 4.
LPT1 or PRN	The first parallel printer port (output only).
LPT*n*	The *n*th parallel printer port up to 3.
NUL	A nonexistent (dummy) device used for discarding output when device output is unwanted. Often used in batch files to suppress (throw away) screen output.

Because these resident device drivers are defined by DOS, their names are reserved. DOS first tests file parameters to see whether the parameters are device names. If you use a file name, even with an extension, that has the root name of a reserved device, DOS assumes that you are referring to the device, not to a file. To avoid such naming conflicts, don't provide file names that are similar to reserved device names. The file name DON is just one keystroke away from the device name CON. In a COPY command line, typing **C** instead of **D** can send file output to your screen, not to DON.

Installable Device Drivers

Cue:
Installable device drivers are loaded from CONFIG.SYS.

Since V2.0 your PC's device complement does not have to be limited to those devices supported by IO.SYS. DOS provides for additional device support to be added at boot time. Figure 4.3 shows the installable device drivers as an extension of the kernel's available device drivers. DOS inspects the contents of the special file CONFIG.SYS in the boot disk's root. If CONFIG.SYS contains any device installation directives, DOS will install the indicated device driver from the indicated disk. Figure 4.3 shows CONFIG.SYS as a pathway from the disk file device drivers to their installed locations in the system. The broken line dividing the device driver box indicates that the number of installed device drivers varies based on CONFIG.SYS. A more detailed description of CONFIG.SYS appears in the next section. By providing a means to load additional device support at boot time, DOS makes the kernel interface to the hardware much more versatile than the one provided for by IO.SYS.

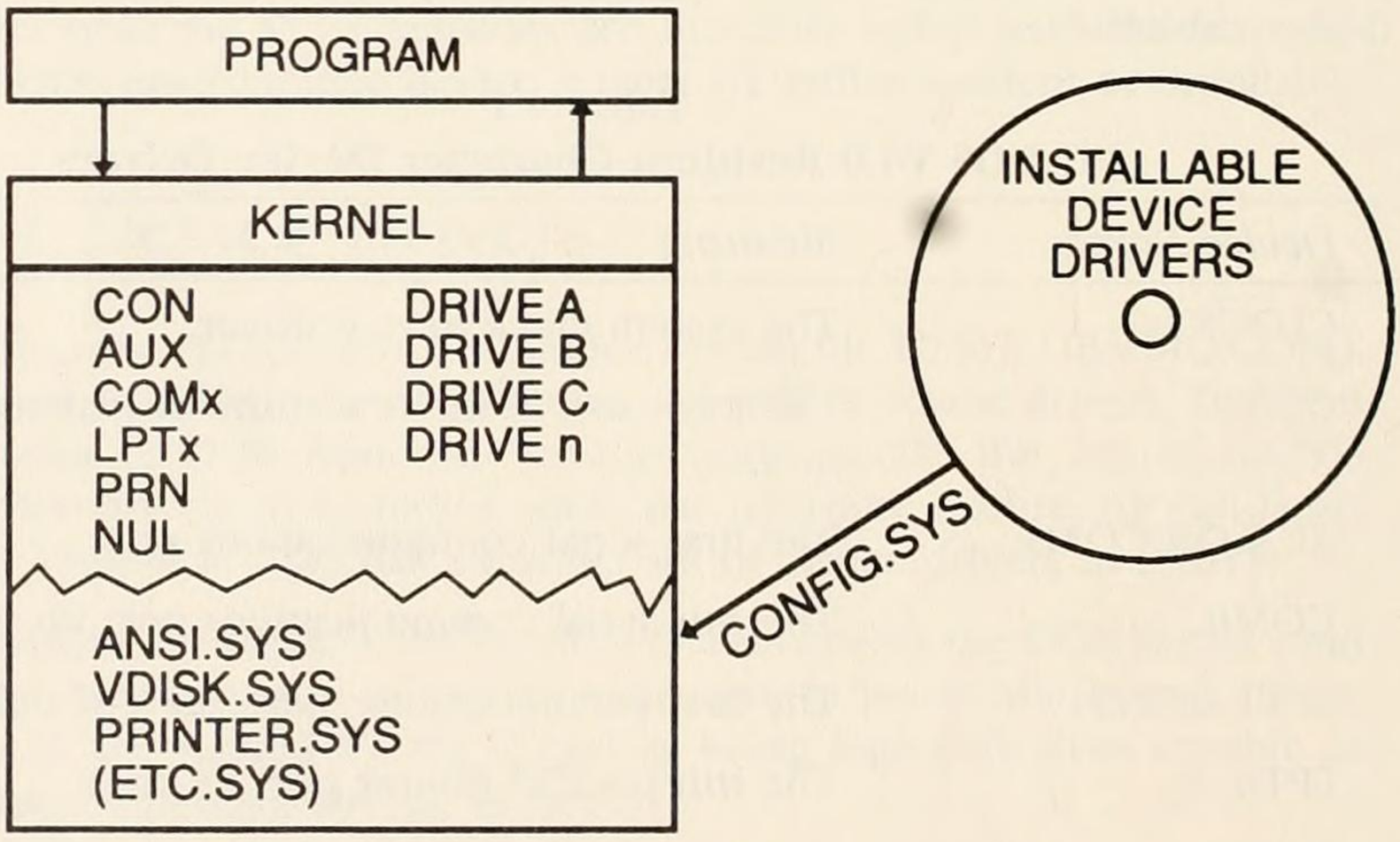


Fig. 4.3
CONFIG.SYS loading installable device drivers.

DOS provides some installable device drivers on the master DOS disks. Again, different versions supply different numbers of device driver files. Table 4.2 shows the installable device drivers that come with PC DOS V4.0.

Table 4.2
PC DOS V4.0 Installable Device Drivers

Device Driver	*Action*
ANSI.SYS	Replaces the standard console device driver with a device driver that incorporates ANSI terminal control sequences
DISPLAY.SYS	Enables code-page switching on an EGA display and on the IBM PC Convertible LCD display
DRIVER.SYS	Operates an external disk drive through a block device letter name
VDISK.SYS	Installs and operates a virtual (RAM) disk device as a simulated disk drive
XMAEM.SYS	Uses extended memory on 80386-based PCs to simulate an expanded-memory adapter device
XMA2EMS.SYS	Supports the LIM expanded-memory specification (EMS) under DOS V4.0

If you look at the action of each installable device driver, you'll notice that only DRIVER.SYS is able to support additional hardware on the PC. DRIVER.SYS can be configured to support an additional disk device that does not conform to the design of the disk devices supported by the resident disk driver routines.

The other installable device drivers in table 4.2 don't add any new devices to the computer. Instead, the rest of the device drivers modify or enhance the resident certain drivers and add logical devices created from existing system resources. Users who don't need the standard devices enhanced don't need to install the device driver that enhances. Some of the installable device drivers are for special purposes.

XMAEM.SYS, for instance, is useful for owners of 80386 CPU-equipped IBM PS/2 computers like Model 80. Through the addressing modes of the CPU and the device driver translation, RAM that surpasses the 1M addressing range of DOS appears to be an expanded-memory adapter board. Expanded and extended memory are covered in a later section of this chapter. Although some installable device drivers included with DOS are for special purposes, others have a more general appeal to users.

XMA2EMS.SYS is another special-purpose device driver that makes an expanded-memory adapter usable with your computer so that DOS or an application may access expanded memory. On an 80386 computer with XMAEM.SYS installed, XMA2EMS.SYS enables DOS or an application to access the memory that is acting as expanded memory. This is an example of two device drivers working together.

ANSI.SYS, for instance, replaces the resident keyboard device and display device driver. ANSI.SYS enables the device's character stream to encode device control sequences that position the cursor, change the screen colors, give keys new definitions, and more. ANSI.SYS is such a useful installable device driver that Chapter 12 is devoted to its use.

VDISK.SYS and MS-DOS's RAMDISK.SYS add one or more virtual drives or RAM disks to a system. Because RAM disks are actually system memory and not a mechanical disk drive, they are very fast. Many users create RAM disks to speed operations involving heavy file searching, such as the spell-checking of a document from a dictionary file.

DISPLAY.SYS and PRINTER.SYS accommodate the international-character-set configurability of DOS. V3.3 and V4.0 have extensive provisions for international configuration. Chapter 15 explains how to use the international configuration considerations of DOS.

Third-party software vendors and hardware manufacturers offer a wide variety of installable device drivers that either enhance existing hardware or accommodate new peripheral hardware. DOS can install these device drivers at boot time, also. The key to installing device drivers lies in CONFIG.SYS. Pointing devices such as mice and digitizer pads often include device driver files. MOUSE.SYS and KOALA.SYS are examples of pointing device drivers. External add-on disk drives often come with special block device drivers. The 5 1/4-inch add-on floppy drive from Sysgen for IBM PS/2 systems is delivered with a driver named BRIDGE.DRV. Note that the DRV extension is used for the disk drive's driver file name. DOS V3 can use other driver file extensions besides SYS. IBMCACHE.SYS is a disk-caching driver that uses extended memory in PS/2 computers to speed disk operations. IBM supplies IBMCACHE.SYS with some of IBM's computers. The variety of third-party installable device drivers is great. Chances are good that you use or will use a third-party installable device driver in your system. Thanks to DOS's use of CONFIG.SYS, installing a device driver is easy.

Understanding CONFIG.SYS

CONFIG.SYS is an optional file that DOS looks for during system booting. DOS does not come packaged with a CONFIG.SYS file in most implementations. You create CONFIG.SYS by using a COPY CON command line or by using EDLIN or another text editor. Part IV of this book covers CONFIG.SYS and the configur-

able aspects of your PC. You will want to read Part IV before you change your CONFIG.SYS file or add a new one. For now, you'll get a brief overview. To see how CONFIG.SYS is used, take a moment to review the boot process.

When you turn your PC's power on for a cold boot or press the Ctrl-Alt-Del key combination for a warm boot, DOS loads itself from the boot disk with the aid of the bootstrap ROM and the boot disk's bootstrap loader routine. The DOS kernel and the BIOS extensions are loaded into memory from the MSDOS.SYS (IBMDOS.COM) and IO.SYS (IBMBIO.COM) disk files. The system is initialized, and DOS looks for CONFIG.SYS in the root directory of the boot disk. DOS will not look in any other directory on any other disk for CONFIG.SYS. Assuming that DOS finds CONFIG.SYS, DOS opens the file and executes the directives found in the file.

DOS looks for configuration commands and their parameters as the directives to carry out. DOS may encounter directives like these:

```
FILES = 15
BUFFERS = 20
```

The first of the two directives tells DOS how many files to accommodate at any time. DOS will set the number of files to 8 if CONFIG.SYS is missing. The default 8 files accommodate DOS's 5 resident character devices with room to open 3 additional files. The second of the two directives tells DOS how many RAM buffers to set aside out of memory for temporarily housing transient I/O data. These two directives are not device driver directives, but they do affect the operation of data interchange between the disk device drivers and the kernel's transient data buffers.

You may also see a line in the CONFIG.SYS file that looks similar to the following:

```
SHELL = C:\COMMAND.COM /P
```

This line tells DOS where to find COMMAND.COM or another command processor. Normally, COMMAND.COM is the command processor, although PCs used in unusual applications may have non-DOS command processors. In all likelihood, your command processor will be COMMAND.COM.

You may also see several lines that contain the DEVICE directive. These lines look similar to the following:

```
DEVICE = ANSI.SYS
DEVICE = MOUSE.SYS
DEVICE = VDISK.SYS
```

DOS looks for these lines in order to install device drivers during the boot process. When DOS encounters a DEVICE = line in CONFIG.SYS, DOS finds the file named in the assignment and installs the content of the file as a device driver. When the installable device driver is in RAM, DOS adds the driver to its list of available drivers. The driver is then available for use.

When CONFIG.SYS is processed completely, DOS completes the boot process by loading COMMAND.COM, finding and executing commands in AUTOEXEC.BAT, and issuing the command line DOS prompt. The absence of AUTOEXEC.BAT causes DOS to prompt for the time and date and then display the DOS prompt.

If you use a hard disk as your boot disk, don't forget to include necessary CONFIG.SYS directives in your bootable working DOS copy. In most cases, you'll want the same device configuration when booting from a floppy as when you boot from your hard disk. On the other hand, you can select for a boot floppy CONFIG.SYS directives that install a varying set of devices from your normal set. Booting from a floppy with a specially prepared CONFIG.SYS file gives you device flexibility without requiring that you edit the CONFIG.SYS file in the root of your hard disk.

Understanding Extended and Expanded Memory

The presence of the DOS V4 XMAEM.SYS and XMA2EMS.SYS device drivers serves to show that DOS is "modernizing" to keep up with the developments in memory-addressing capabilities of the 80286 and 80386 members of the Intel 8086 microprocessor family, on which DOS is based. DOS can address only 1M of memory in a normal way. The 1M limit is due to the number of address lines available on the 8086 microprocessor. The 8086 has 20 binary address lines. Because the binary condition consists of 2 states, 2 to the 20th power is 1,048,576, or 1M. To address just one byte more of data than the 1M of data that 20 address lines support, a microprocessor must have 21 or more lines. When DOS was designed, the Intel microprocessors didn't have 21 or more lines. The later 80286 and 80386 do have more than 20 address lines and can address far more memory than 1M.

Programs and data reside in memory. The more memory a PC has, the more program size and data the PC should be able to accommodate. Without some electronic intervention, this larger-memory, larger-capacity axiom is not true for DOS. When DOS was designed, the 80286 and 80386 processor chips were not used in PCs. The 8086 and 8088 chips with their memory limit of 1M were the top Intel microprocessors of the era. DOS designers placed DOS routines at specific locations within the 1M of memory space available. DOS's internal tables could be used to manipulate addresses within 1M.

When the 80286 and 80386 microprocessors began to appear in PCs, DOS already had a huge installed base. To accommodate the larger memory capacity of the newer CPUs, DOS would have had to abandon the installed base of users and program developers. DOS designers chose not to change the basic arrange-

ment of DOS and to keep DOS uniform with the installed base. Still, the lure of larger memory capacities gave third-party vendors the incentive to find a way for DOS to break the 1M memory barrier.

Two basic methods are used to give programs access to more memory. The first method is the use of extended-memory addressing. The name given to the memory involved in extended-memory addressing is, not surprisingly, *extended memory*. The second method that evolved uses memory through a device driver. The additional memory is not an extension of the existing memory below 1M. The device-accessed memory is actually an expansion of the memory below 1M. The memory involved in the device access method is called *expanded memory*. Expanded memory follows the Expanded Memory Specification, or EMS. Because *extended* and *expanded* are similar terms, it's easy to confuse the two. Extended and expanded memory are part of today's PC life, however, so you should understand the difference between them.

Understanding Extended Memory

The Intel 80286 has 24 address lines and can address 16,777,266 locations, or 16M, of memory. The newer 80386 and 80486 have 32 address lines and can address 4,294,967,296 locations, or 4G (gigabytes), of memory. The first 1M of this address range addresses the system's conventional memory. The first 1M is called *conventional memory* because the 8086 chip uses only 1M and because DOS is designed around this 1M convention. Addresses above the 1M point are called *extended-memory addresses*. Figure 4.4 shows the addressable memory of the various Intel 8086 family members. The figure shows that the 80286 and 80386 can address a significant number of extended addresses.

Any memory at these extended addresses is called extended memory. Only the 80286, 80386, and 80486 can use extended memory. The 8086 and 8088 cannot.

Cue:
The 80286, 80386, and 80486 can use extended memory.

The 80286, 80386, and 80486 address conventional memory like an 8086 by using an internal operating mode called *real* mode. In real mode the microprocessors are restricted to the same 1M address range as the 8086. Unlike the 8086, however, the newer microprocessors can enter another mode to address the extended-memory locations. This extended-memory mode is called *protected* mode. DOS became a standard before the advent of protected mode and can't effectively take advantage of protected mode's extensive address range. Figure 4.5 shows how conventional memory is divided. DOS addresses the first 640K of memory as conventional low memory. DOS addresses the remaining 384K of the 1M conventional memory as high conventional memory.

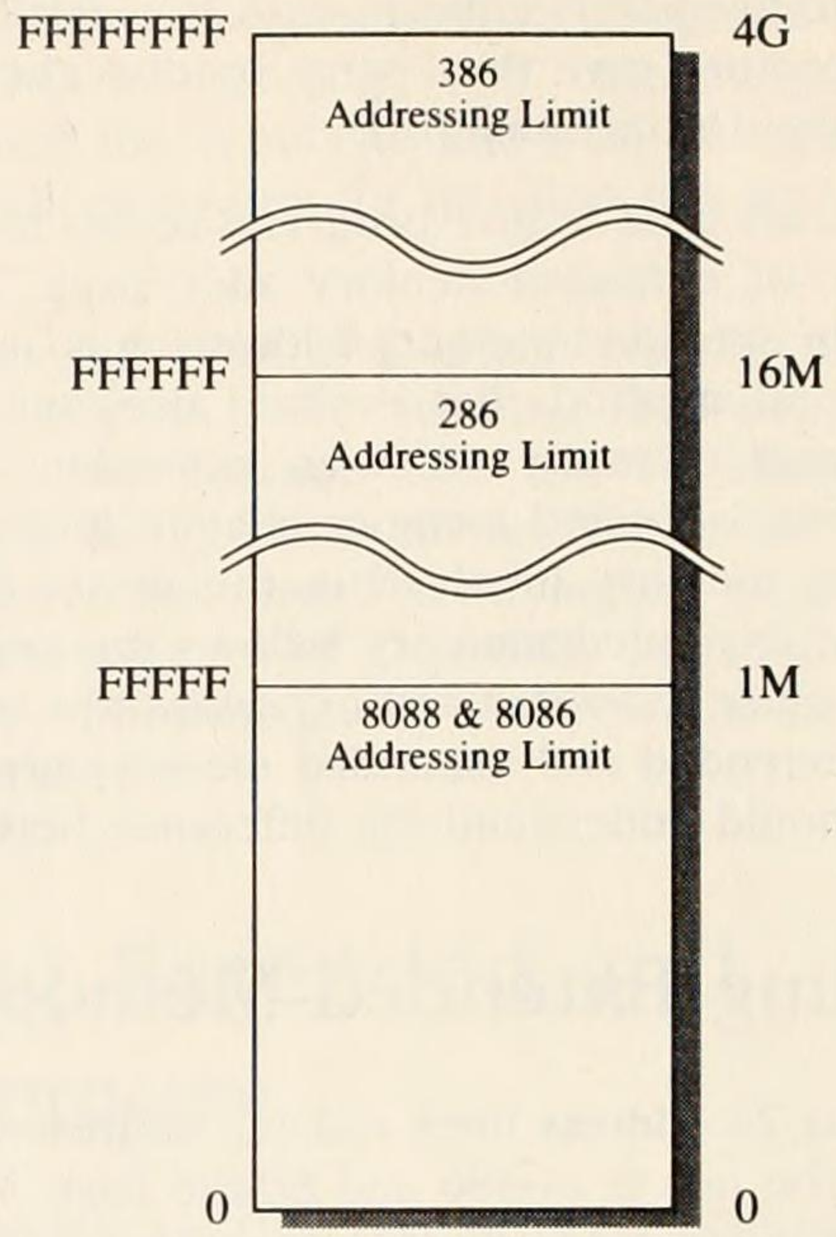


Fig. 4.4

A map of addressable memory by the various CPUs. The 386 addresses 4,000 times more memory than the 8088/8086. (The map is not drawn to scale.)

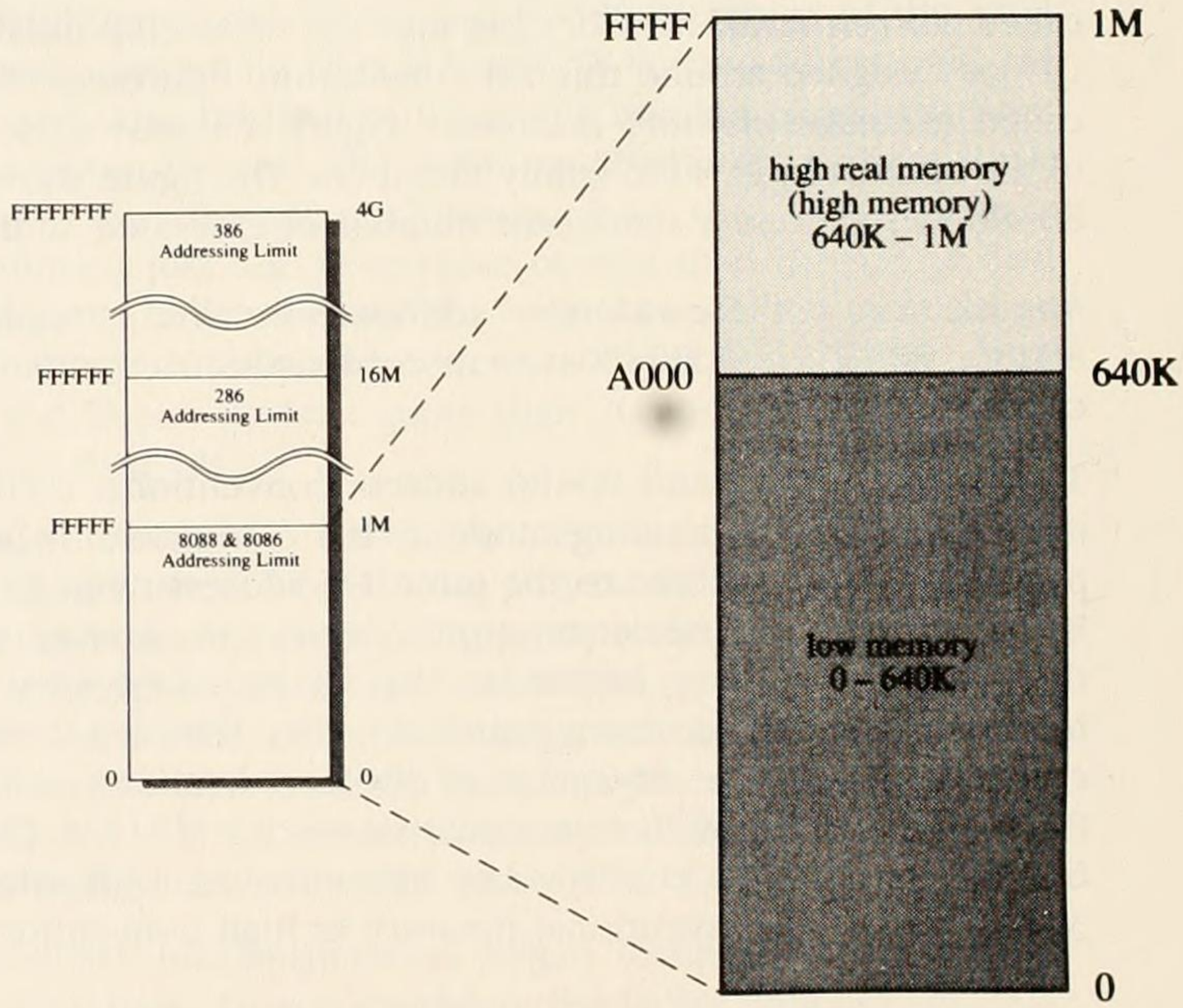


Fig. 4.5

The first 1 megabyte of memory (real memory) in all personal computers is broken into two segments. Low memory, from 0K to 640K, is devoted to user RAM. High memory (from 640K to 1M) is reserved for adapters and system ROM. For 286-based systems, a computer with 1M of RAM is broken into 640K of low memory and 384K of extended memory.

Some DOS utility programs and drivers can use extended memory, however. The programs switch the CPU from real mode to protected mode. The DOS kernel has nothing to do with the switch. Because the kernel is DOS's administrative bookkeeper, the kernel is unaware of how the extended memory is being used or what programs are using it. The programs that use extended memory in protected mode must indicate to each other, through memory-based tables, which program is using which memory. Unless the programs agree to "cooperate," any program can "steal" the extended-address range of another program. The result is an extended-memory conflict. The DOS device drivers that can use extended memory cooperate in memory use, but "outside" programs that use extended memory may not.

Protected mode and extended memory have some operational disadvantages when used with a real-mode operating system such as DOS. The switch from real to protected mode is not easy for a programmer to code. The DOS kernel and the BIOS offer little functional support for protected mode. Programs that use extended memory must do some careful checking in set-aside memory locations to determine which programs are using which extended memory. Unfortunately, not all extended-memory programs check thoroughly, and memory conflicts arise. Some memory conflicts simply make a program work erratically. Some conflicts "crash" a program so badly that you have to reboot. Still, the lure of additional memory beyond the limits of 1M keeps programmers trying the extended-memory protected-mode method. Another method of accessing more memory exists that enables even the 8086 to take advantage of more RAM: the expanded-memory method.

Understanding Expanded Memory

Although the term *expanded* sounds very much like *extended*, expanded memory is not extended memory. Programs, including some DOS programs, access the contents of expanded memory, or *EMS,* in real mode as addresses within the first 1M of memory. Using expanded memory, any CPU in the 8086 family can break the 1M conventional-memory barrier. The way expanded memory provides access to memory that is additional to conventional memory marks the true difference between expanded memory and extended memory.

Expanded memory is accessed through a device driver as though the expanded memory were a device. A defined portion of the expanded memory is temporarily assigned to a predefined portion of conventional memory. This defined portion of expanded memory is called a *page*. The predefined portion of conventional memory is called a *page frame*. Figure 4.6 illustrates the paged-memory concept. Notice that more than one memory page is associated with a page frame.

Cue:
Expanded memory is divided into portions called pages.

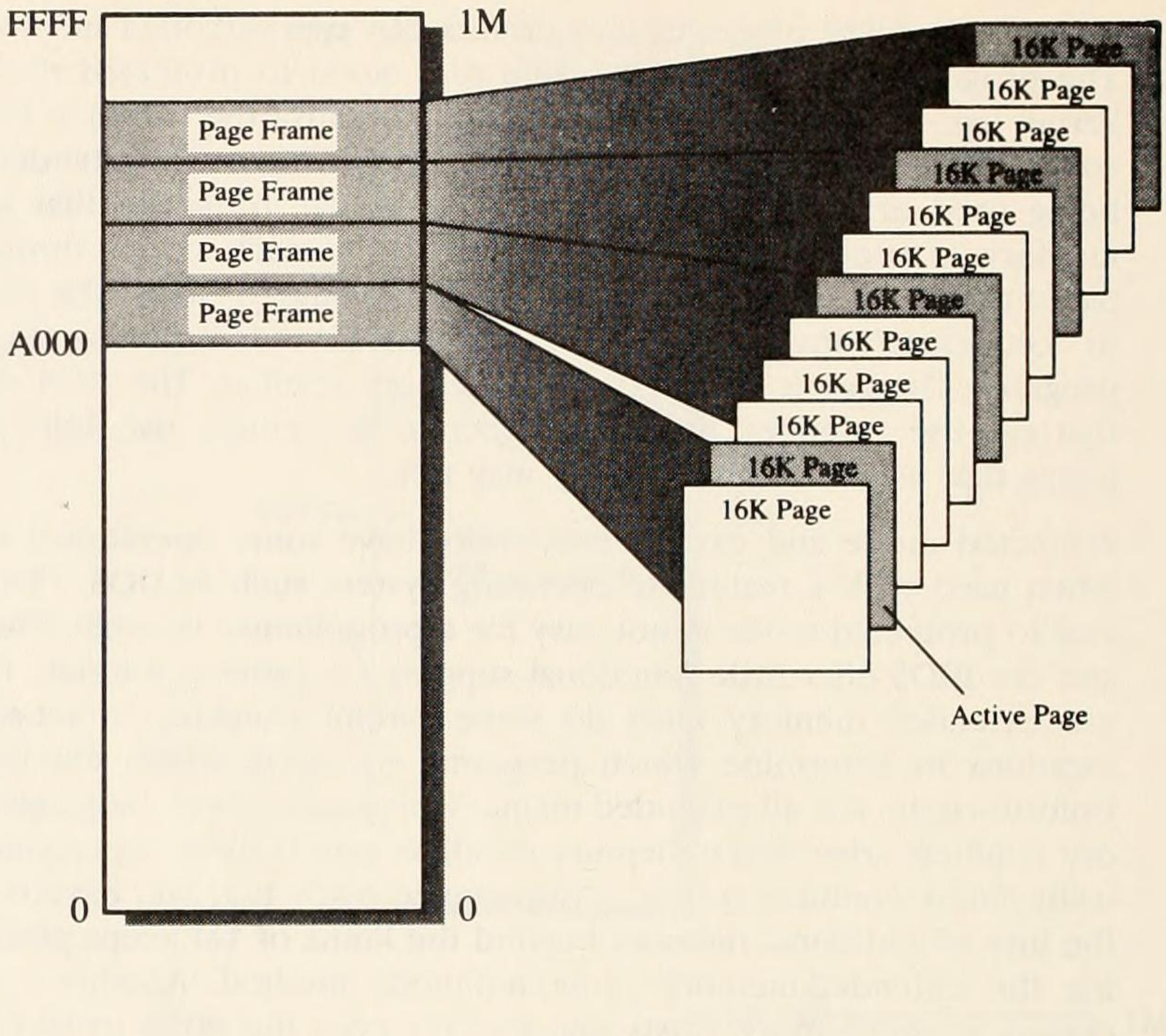


Fig. 4.6

EMS memory works by swapping 16K pages of physical memory on the EMS board into page frames in conventional memory. Page frames are usually located in empty 16K spaces in high real memory. When exposed through the page frame, the physical page is accessible by the CPU as conventional memory. A page frame can use any EMS page, but only one page at a time.

When the CPU addresses the page frame, the CPU is actually accessing the values in a page of expanded memory. The expanded-memory device driver locates unused portions of conventional memory in 16K blocks and uses the blocks for page frames. Normally, the number of 16k pages of expanded memory on the expanded-memory adapter far exceeds the number of page frames established in conventional memory by the driver. To access all the pages of expanded memory, the driver must swap pages of memory in and out of the available page frames.

When a program needs to access a page of expanded memory that is not in the page frame, the device driver must establish the new page in the page frame. This "swapping" of page frames is a processing overhead that extended-mode addressing does not have. Still, expanded-memory operation offers the capability for specially designed programs to have access to more than 1M of memory through page frames. Expanded memory also has the advantage of following recognized standards that make memory conflicts rare.

Cue:
EMS comes in two versions: LIM 3.2 and LIM 4.0.

Two different versions of EMS exist. LIM 3.2 and LIM 4.0 both use the paged switching of expanded memory. LIM 4.0 provides for more flexible page-frame location and greater expanded-memory access. If you have expanded memory on your system, check your system documentation to determine whether it is LIM 3.2 or LIM 4.0. Applications software that uses expanded memory is usually configurable to either type. Because of the additional flexibility, LIM 4.0 is more desirable.

Computers that have the 80386 or 80486 processor can take advantage of the advanced memory-management features of the processor and can thus emulate an expanded-memory adapter board. DOS V4 provides XMAEM.SYS and XMA2EMS.SYS to "trick" extended memory into following the EMS convention. These two device drivers create a logical bridge between extended-memory- and expanded-memory-capable programs. Chapter 14 discusses their use.

Understanding Standard Input and Output

DOS commands use the CON device as their standard command-input and message-output device. Some commands prompt you for information. Others just wait for you to supply information; you supply this information in the form of a keyboard response. Supplying information to a DOS command is an example of supplying standard input. Take, for example, the SORT command. SORT takes lines of input and then outputs the lines in an alphabetically sorted order. SORT acts on the input stream like a specialized filter. In fact, SORT is one of DOS's special filter commands. You issue the command as the following:

```
SORT
```

When you issue the command, the cursor drops to the next line, and SORT waits for standard input from the keyboard. You provide input such as the following:

```
BRAVO
CHARLIE
ALPHA
```

After entering **ALPHA**, you press Ctrl-Z or F6 to tell SORT that you have completed input. SORT alphabetizes the lines that you input and displays them on the standard output device, your screen, as the following:

```
ALPHA
BRAVO
CHARLIE
```

SORT is not very handy when you have to provide the standard input from the keyboard. Fortunately for SORT (and you), DOS's treatment of devices enables standard input and output to be redirected from and to other devices. The input for SORT can come from a file or another command's output; the output of SORT can go to a file or another command's input. The temporary reassignment of inputs and outputs is called *redirection*.

Understanding Redirection

Cue:
the symbol < indicates input to come from a file or device.

Because the DOS kernel processes I/O requests from programs through a uniform set of calls to device drivers, the kernel can view devices as being rather universal. With only minor programmatic intervention, the kernel can accommodate the substitution of a different input device for the standard input device. The same substitution capability applies to an output device. DOS knows when to substitute or redirect input and output by recognizing special characters in the command line syntax. On the command line, the special redirection symbol < tells DOS that input will come from the file or device name that follows the symbol.

Suppose that a text file named TODO.TXT in the working directory contains these lines:

```
03-22-89... Go to doctor's appointment at 3:30
03-17-89... Dinner with Bill
01-01-89... Resolution: Use BACKUP regularly
04-04-89... Buy David's birthday present
03-20-89... Get chemicals for spa
```

These lines obviously make up a reminder file, but the dates are out of order. Thanks to SORT and redirection, you can display them in order. You simply issue the SORT command, using the input redirection symbol to show input from TODO.TXT. The command line looks like this:

SORT <TODO.TXT

SORT alphabetizes the lines and, in doing so, arranges the dates in order. The screen (standard output device) output shows the following:

```
01-01-89... Resolution: Use BACKUP regularly
03-17-89... Dinner with Bill
03-20-89... Get chemicals for spa
03-22-89... Go to doctor's appointment at 3:30
04-04-89... Buy David's birthday present
```

Cue:
The symbols > and >> indicate output to come from a file or device.

The same alphabetized lines can be written back to a file through output redirection. The special > and >> symbols in the command line syntax tell DOS to redirect output to the file or device that follows the output redirection symbols. In the SORT example, the command line syntax looks like this:

SORT <TODO.TXT >>BYDATE.TXT

DOS won't display the sorted lines on the standard output device, the screen. Instead, the output will be redirected to the file named BYDATE.TXT.

The input redirection symbol < simply tells DOS to continue getting input from the input device or file given. The double output redirection symbol >> tells DOS to add the current output to existing output in the device or file shown. Chapter 9 discusses SORT and redirection further. Various command examples in Part II use redirection.

The FIND filter command takes its input from a file or from the keyboard and determines its redirection requirements internally. If you give a file name as a FIND parameter, FIND automatically uses the content of the file instead of using the input from the keyboard. FIND, which looks for a line in the file (or from the keyboard) that matches a quoted text specifier, is useful in the BYDATE.TXT file example. Consider the following command:

FIND "David" BYDATE.TXT

FIND searches the lines in BYDATE.TXT for a set of characters matching "David" (without the quotation marks). FIND outputs the following:

```
---------- TODO1.TXT
     04-04-89... Buy David's birthday present
```

Through redirection, you could send the output of FIND to another file. Redirection makes possible some creative DOS work.

Understanding Piping

DOS's logical way of viewing devices makes possible another form of input and output substitution: piping. *Piping* gets its name from the concept of coupling commands like pieces of pipe, where the command's (pipe's) output is the next command's (pipe's) input. Typically, a DOS filter command is used as one of the commands in the pipe line. DOS filter commands are covered in Chapter 9.

The special symbol | is DOS's indication that two commands are being joined. The output of the first command goes to the input of the second command, rather than to the standard output device. The input of the second command is the output of the first command instead of the output of the standard input device.

An example of a piped command uses the DIR command and the SORT filter. You can take a directory listing and sort the listing alphabetically with the following command:

DIR C:\DOS\DRIVERS | SORT

The directory listing is sent to SORT, which reports the filtered listing as this:

```
      14 File(s)    3186688 bytes free
 Directory of  C:\DOS\DRIVERS
 Volume in drive C is MYFILE
 ANSI     SYS       1678 03-17-87  12:00p
 BRIDGE_B DRV       8056 11-30-88   2:53p
 CON525   SYS        120 10-19-88   1:24p
 CONNO525 SYS         93 11-07-88   8:12a
 COUNTRY  SYS      11285 03-17-87  12:00p
 DISPLAY  SYS      11290 03-17-87  12:00p
 DRIVER   SYS       1196 03-17-87  12:00p
 IBMCACHE SYS       8032 04-02-87  10:00a
 KEYBOARD SYS      19766 03-17-87  12:00p
 MOUSE    SYS      14298 09-28-87  12:00p
 PRINTER  SYS      13590 03-17-87  12:00p
 VDISK    SYS       3455 03-17-87  12:00p
```

The output of the DIR command is reported by SORT as an alphabetized list.

Summary

Understanding how DOS relies on device drivers enables you to see how you can use DOS commands to gain control of your hardware. Although the detailed operation of DOS devices is hidden below the computer's surface, DOS configuration commands, as well as concepts like redirection and piping, give evidence that DOS is controlling devices. DOS, through COMMAND.COM, views devices in a logical way. Programmers using the DOS-defined kernel-to-device-driver interface can ensure that external devices integrate smoothly into the working PC. The following are key points covered in this chapter.

- Device drivers add new hardware or modify the operation of existing hardware.
- DOS uses two types of device drivers: character and block.
- Resident device drivers are loaded during booting from the hidden DOS file IO.SYS (IBMBIO.COM).
- Installable device drivers are optional and are loaded during booting from directives in the CONFIG.SYS file.

- Programs can overcome DOS's 1M real-memory limit by accessing extended memory and expanded memory.
- Extended memory is memory above 1M accessed by the CPU in protected mode.
- Expanded, or EMS, memory is additional memory accessed through page frames. EMS memory is either LIM 3.2 or LIM 4.0.
- Through redirection and pipes, you can change commands' standard inputs and standard outputs. Redirection is most often used with devices; pipes are used to connect commands.

The information in this chapter and the three preceding chapters provides a good framework for understanding DOS. The material in these chapters of Part I covers aspects of DOS's operation that give you a view below DOS's surface. Of course, you don't have to be a software designer to use DOS competently. Having to be that technical would run counter to the spirit of personal computing. But if you understand the material presented in Part I, you will certainly be more confident and capable when using the DOS commands presented in Parts II, III, and IV. You'll even be in a better position to gain insight from Part V's look at software that works with DOS to add operating system value.

The next chapter starts Part II, which covers the core commands of DOS. Chapter 5 covers the disk maintenance commands. With your knowledge of the file system and devices, you are ready to learn more about the functional block of disk-level commands.

Part II

Working with the Core Commands

Preparing and Maintaining Disks
Managing and Navigating the Directory Hierarchy
Managing Your Files
Working with System Information
Gaining Control of Devices

5

Preparing and Maintaining Disks

Chapter 2 presented a short course in the fundamentals of the DOS file system and related topics. This chapter presents the commands that directly relate to the preparation and maintenance of disks. All the commands presented in this chapter can be viewed as disk-level commands; REPLACE, RECOVER, and CHKDSK, however, have file-level options. If you are like most DOS users, you spend more time managing files than you do managing disks, so you are likely to be more comfortable working with file-level DOS commands than you are with disk-level DOS commands. You don't format your hard disk, for example, just for practice. These disk-level commands, however, put you in the driver's seat of disk preparation and maintenance. Knowing when and how to use these commands will enhance your PC competence.

Each disk-level command, its possible parameters, and any optional switches are covered in more detail in the Command Reference section. This chapter provides general information about the disk-level commands, such as how they do their job within DOS and what rules apply in their usage.

The commands covered in this chapter include the following:

- FDISK
- FORMAT
- LABEL
- VOL
- SYS
- DISKCOPY
- DISKCOMP
- REPLACE

- CHKDSK
- RECOVER

Issuing External Commands

Cue: *The commands built into the COMMAND.COM program are the DOS internal commands.*

Because this chapter introduces the first DOS commands of the book, take a moment to review the difference between internal and external commands. Chapter 3 discussed COMMAND.COM and its built-in commands—the internal DOS commands. You can issue an internal command whenever you see the DOS prompt; COMMAND.COM will execute the internal command immediately. External commands, however, require some extra consideration.

Because external DOS commands are program files that COMMAND.COM loads and executes, they must be accessible to the command processor. This section discusses the process of making these commands accessible to COMMAND.COM.

Reminder: *Internal commands can be issued at any time from the DOS prompt.*

As an example of external commands, look at the FDISK command. The full syntax is as follows:

*dc:pathc***FDISK**

dc:pathc\ is the optional full path name, which consists of a drive name and path name on the drive in which the external command is located. You can omit the drive name if the command is on the default drive, and you can omit the path name if it is the current path on the drive that contains the command. You can also issue the command without a full path name if the full path name is included in the latest **SET PATH** *full pathname* command.

If your external DOS command programs reside in a directory named \DOS on drive C:, you can issue the following command:

PATH=C:\DOS

Reminder: *External DOS commands are program files that must be loaded and executed by COMMAND.COM.*

DOS will search the \DOS directory on drive C: if the external command is not in the current directory. If you have not yet used the PATH command, review the information on PATH in Chapter 6 before you continue with this chapter. Your computer may have an AUTOEXEC.BAT file that sets the path automatically.

Dividing a Hard Disk with FDISK

The hard disks used by today's PCs can be logically divided into partitions. (See Chapter 2 for more information on partitions.) A partition is simply a section of the storage area of a hard disk. Many operating systems, including DOS, can

utilize disk partitions, and most have some utility program that creates and manages partitions. DOS's utility program for partition creation and management is the external command FDISK. The details and operational considerations of FDISK are presented in Appendix D. Use the presentation in Appendix D when you need to use FDISK; the Appendix contains step-by-step instructions for using the command with different versions of DOS. This chapter's view of FDISK provides a general view of the command.

Cue:
The FDISK command creates and manages disk partitions.

Issuing the FDISK Command

The syntax for FDISK is as follows:

FDISK

Notice that there are no parameters for FDISK. Because FDISK handles the partitions of any hard disks on your system in a sequential fashion, there is no need for a drive parameter.

Understanding Hard Disk Partitions

The concept of dividing one real or physical disk into more than one apparent or logical disk drive can be traced back to pre-DOS operating systems. In earlier minicomputer disk operating systems, a system nucleus file included the provisions to translate head and cylinder boundaries of the popular disk drives of the day into logical disk drives. If a minicomputer system had one 32M removable platter disk drive and one 64M hard disk, the size differential of the real disks precluded the use of disk-level copying utilities.

If the system nucleus file of the operating system included a table that split the storage space of the 64M disk in half, the system could be tricked into thinking that it had three 32M disk drives. Copying from any 32M "drive" on the system to any other was therefore possible. The operating system wasn't really tricked, though. During the boot process of the minicomputer, the operating system would load the table into memory reserved for data-file management. Each time a program made a disk request, the disk address (head and cylinder numbers) for the file on the logical 32M disk volume was intercepted by the data-file manager section of the operating system. The data-file management system would use the table to transform the logical disk address into a real or absolute disk address. The applications program and the higher levels of the operating system were no more the wiser to the switch. Because no operating system was dominant across computer manufacturers' lines, most manufacturers added their own innovations to the process of creating logical disks out of a physical one.

Cue:
V2.0 was the first DOS version to support hard disk partitioning.

DOS started out supporting floppy disks only, so prior to V2.0, DOS did not need to address hard disk partitioning. As hard disk drive innovation and competition drove hard drive prices lower, however, the IBM PC/XT with its 10M hard disk and DOS V2.0 brought DOS out of the floppy disk category and into the high capacity storage category. With over 25 times the storage of the 360K floppy, the PC/XT's 10M disk was an ideal candidate for using an additional operating system such as Microsoft's UNIX-derived XENIX. From a capacity standpoint, the XT has no problem accommodating DOS and XENIX on the same disk. The real problem is that the two file systems are incompatible. The obvious solution is to partition the disk into two parts.

Microsoft and other operating system suppliers uniformly left a reserved area of the hard disk available for the partition table's data. Operating system vendors understood that a user might want to install an additional operating system on the hard disk. Most operating systems are not compatible with other operating systems. An operating system, such as DOS, wants to establish the disk as its own. Fortunately, most operating systems consult the hard disk's partition table when booting from the hard disk. The partition table establishes the boundaries of the physical disk that each operating system can call its own.

Chapter 2 describes the disk-level boot record and bookkeeping tables that DOS places in track 0. If another operating system "owns" track 0 as part of the other operating system's partition, DOS cannot use track 0 without damaging the internal tables of the other operating system. DOS avoids the problem by consulting the partition table at the time of start-up. If DOS does not "own" track 0, DOS simply uses the first track that it does own as its track 0. DOS therefore translates the first physical cylinder of its partition into a logical first cylinder of a disk. DOS also considers the last physical cylinder of the DOS partition as being the logical end of the hard disk. Other operating systems use the partition table information to "peacefully coexist" on the same physical disk with another operating system.

The partition table is therefore a concept that supports multioperating systems. With DOS V2.0, the concept of the DOS partition was born. The tool that created and maintained the DOS partition was FDISK.

Understanding the Operation of FDISK

Caution:
To install another operating system on your hard disk, use the instructions supplied with FDISK or the similar command of the other operating system.

The primary function of the FDISK program is to provide the DOS user with high-level access to the disk's partition table. FDISK allows DOS to participate in a multioperating system disk-sharing arrangement. Other operating systems, such as XENIX, have their own versions of FDISK to manage the partition table. You can use FDISK to create a DOS partition of the size you specify, to make the DOS (or another operating system's) partition active, and to delete an existing DOS partition. You can also use FDISK to define partitions for up to four different operating systems.

NOTE: If you have another operating system to install on your hard disk, use the instructions supplied with the FDISK or similar command of the other operating system.

With DOS V3.3, FDISK can be used to create more than one DOS partition. After you define a *primary* DOS partition with FDISK, you can define an *extended* DOS partition which you divide into one or more logical drives. All primary DOS partitions and logical drives must be properly formatted using the FORMAT command before they can be used. When DOS is the active operating system as reflected in the partition table, DOS will boot from the active DOS partition. You can then log to one of the logical drives. The primary DOS partition is normally assigned the drive name C:. The extended DOS partition can be divided further by FDISK into additional logical drives.

Cue:
You can use FDISK V3.3 and later versions to create more than one DOS partition: a primary DOS partition and an extended DOS partition.

Before you format a new hard disk, you must ensure that the disk has been prepared with FDISK. DOS cannot work with a hard disk that has no DOS values in its partition table. In fact, if you have in your computer system another physical hard disk that has been partitioned, DOS will move the drive letter that would apply to the unpartitioned drive to the next physical drive's primary DOS partition. Remember that DOS uses drive letters as names for drives in a multidisk system. This is an important point, so note the following example.

Caution:
Before you format a new hard disk, you must ensure that the disk has been prepared with FDISK.

Suppose that you have two hard disks in your system with one DOS partition each. Their DOS-assigned drive names are C: and D:. Suppose that drive C: fails and you replace it with a new unpartitioned drive in the failed drive's position. DOS will detect that the new drive lacks a DOS partition and will assume that the second hard disk is drive C: or the next usable disk in line. Because you are used to the first hard disk as C: and the second hard disk as D:, you may decide to format the new first drive with the command **FORMAT C:/V/S**. If you don't catch the mistake at the warning prompt, DOS will format the second hard disk that you refer to as D:. The result is disaster. The second hard disk is wiped out because the first hard disk was not partitioned. Remember that the presence or absence of primary or extended DOS partitions will affect the drive letters that DOS assigns to physical or logical drives. The drive-name movement can also occur when you delete an extended partition or logical drive.

Caution:
The presence or absence of primary or extended DOS partitions will affect the drive letters that DOS assigns to physical or logical drives.

Once you partition your hard disk(s), DOS will respect the logical drives and create a file system for each drive when it is formatted. If you use FDISK to delete a partition and subsequently create another partition, all the data in the original partition will be destroyed. Don't worry about FDISK doing something rash when you run it. You will see warnings when you are electing to do something with FDISK that destroys data. In order to proceed beyond the warnings, you must answer a confirmation prompt.

Before FDISK deletes a partition, for example, you will see the following warning message:

```
Warning: Data in the DOS partition
will be lost. Do you wish to
continue (Y/N). . . . . ? [N]
```

Respond N if you do not want to continue with the partition deletion operation. Respond Y if you are sure that deleting the partition is your intention.

Because of its potentially dangerous effects, FDISK is one DOS command you should not practice with often. There are, however, some opportunities to work with FDISK. If you decide to reformat your disk, for example, and you have one DOS partition on the whole disk, you can run FDISK first and experiment. If you are upgrading to a hard disk from a floppy-only system, you can experiment with FDISK before you make your "official" partition decisions. Just remember that you will need a bootable floppy disk that contains the FDISK and FORMAT programs. As always, you should have a working copy of your DOS master system disks. Deleting partitions can cause you to rely on the system's floppy disk until the hard disk is again partitioned and formatted.

Formatting Disks with FORMAT

Before disks can receive files, they must be initialized by the external DOS command FORMAT. From a user's point of view, formatting a disk consists of issuing a properly constructed FORMAT command and letting the computer do all the work. Actually, after running FDISK on a hard disk, FORMAT is the most essential disk-level command. In fact, the *only* thing that DOS can do with an unformatted disk is format it!

Caution: *FORMAT destroys all files and directories on a disk.*

Before you read on, remember the following caution: FORMAT destroys all files and directories on a disk!

Always use care and caution when formatting disks. Although special programs that can recover some formatted disks to their preformatted state are available, you should consider FORMAT a dangerous command to misuse.

The Command Reference section of this book covers the FORMAT command thoroughly; the following sections provide insight and examples for your use of FORMAT.

Issuing the FORMAT Command

The syntax for FORMAT is as follows:

FORMAT d: */switches*

d: is the drive which holds the disk you want to format; */switches* are one or more of the optional switches listed in table 5.1.

The Command Reference section provides more detail on allowable switch combinations and which switches are available with different versions of DOS.

Table 5.1
FORMAT Switches

Switch	*Action*
/1	Formats a diskette as a single-sided disk. /1 is often used with the /8 switch to format 8-sector-per-track diskettes to be compatible with V1.0 and V1.1 systems.
/3	Formats a 3 1/2-inch 720K diskette in a 1.44M drive.
/4	Formats a 5 1/4-inch diskette with 40 tracks per side (40 cylinders) in a HC (1.2M) drive. The resulting disk may be unreadable by a standard 360K (double-sided, double-density) drive.
/8	Formats a diskette with 8 sectors per track instead of 9 or 15.
/N:nn	Formats a diskette with the number of sectors specified by *nn*. The disk drive, controller electronics, and the driver must support the specified number of sectors.
/S	Produces a bootable DOS system disk that contains the hidden DOS system files and COMMAND.COM.
/T:nn	Formats a diskette to the number of tracks (per side) that you specify in *nn*.
/V	Causes FORMAT to prompt for a volume label of up to 11 characters. When you have entered the volume label, press Enter.
/V:volume label	Causes FORMAT to label the disk being formatted with the volume label in *volume label*. This version of the /V switch is new with V4.

Understanding the Operation of FORMAT

Cue:
The FORMAT command prepares a disk to accept files—a process known as initialization.

FORMAT destroys the existing data on a used disk. That statement is a precaution to let you know the potentially dangerous side of the FORMAT command. From the standpoint of disk management, FORMAT *initializes* a disk. In other words, it prepares a disk to accept files. FORMAT prepares a disk for the maximum capacity in tracks and sectors that its drive will support (unless optional switches included with the FORMAT command indicate otherwise). DOS knows the number of heads, cylinders, sectors per track, and bytes per sector from the drive type. DOS also knows the number of sectors per cluster and the size in sectors of the root directory and the FAT (see Chapter 2 for more information on the FAT).

FORMAT divides its work between two main tasks. First, FORMAT records the DOS bookkeeping information on the disk in a control area; FORMAT then writes "dummy" data into the remaining part of the disk (called the *files* area). If you use the optional /V and /S switches, FORMAT will additionally write for the disk a volume label you designate and copy the hidden system files along with COMMAND.COM to make the disk bootable. (V4 FORMAT prompts for a volume label automatically and creates a unique disk serial number also.) The following sections look at FORMAT's operation in the control area and the files area. The disk's volume label and system files are also covered.

Formatting the Control Area

DOS, through FORMAT, reserves some early sectors of every disk for recording information it needs to manage the disk. These sectors (usually track 0) are collectively called the *control area* of the disk.

Cue:
Every DOS disk has a boot record.

The first sector on the disk (or partition) is used for the *boot record*. The boot record tells DOS the size and number of FATs that the disk contains and the number of entries that the disk's root directory will accommodate. Details about a disk's heads, sector number, sectors per cluster, and other details that give the disk type its unique BIOS "fingerprint" are recorded here. The boot record contains a brief low-level program that allows the BIOS to begin loading DOS into the system. Every DOS disk has a boot record even if that disk is not bootable. As you will recall from Chapter 3, a disk's DOS bootability is actually determined by the presence of the DOS hidden system files and COMMAND.COM on the disk.

After creating the boot record, FORMAT creates the primary and secondary file allocation tables—the FATs. (See Chapter 2 for an extended discussion of the FAT.) To review the concept of the FAT, remember that every cluster on the disk has an entry in the FAT and that the amount of disk space given to the FAT depends on the capacity of the disk. Generally, larger capacity disks receive more space for a FAT because they have more clusters.

DOS uses the FAT to allocate disk space and to create a traceable chain of cluster locations for each file. A file's first cluster is entered into the file's directory entry. The FAT's cluster allocation unit consists of one or more sectors. DOS can extract sector numbers from cluster numbers by using the sector-to-cluster ratio as given in the boot record. DOS deals with all disk drives on a sector-number access basis and relies on lower-level support routines to translate DOS's relative sector numbers to cylinder, head, and track coordinates. When FORMAT creates a FAT, all files added to the disk through FAT allocation will be located in contiguous sectors on the disk. As files are deleted and new files take their place, however, the FAT entries and the files on the disk become fragmented. You can test for fragmentation by using CHKDSK, a command which will be covered later in this chapter.

Reminder: *DOS uses the FAT to allocate disk space and to create a traceable chain of cluster locations for each file.*

Fragmentation occurs during normal file creation and deletion as a result of DOS's first-available, next-used file allocation method. A fragmented file may be physically spread across a disk in noncontiguous blocks of clusters. DOS has no problem gathering all of the blocks of a fragmented file. Fragmentation does, however, slow file input/output; the disk needs to move its heads to the various locations of the fragmented blocks in order to supply DOS with the entire file.

Reminder: *Fragmentation occurs during normal file creation and deletion*

Managing Disks Larger than 32M

Until the introduction of V4, DOS used 16 bits (binary digits) to represent sector numbers internally. In binary notation, 16^2 is 65,536. In order to use the 16-bit sector numbers, DOS must hold the number of sectors on a disk to 65,536 or fewer. Because all versions of DOS have used 512-byte sectors, the largest drive capacity (or partition) DOS could manage was 512 bytes for each of 65,536 (64K) sectors or 32M. By introducing the extended partition in V3.3, DOS allowed an extended partition of more than 32M, even though no logical drives of the V3.3 extended partition can be larger than 32M.

Cue: *DOS V4 allows a disk or partition to be larger than 32M.*

DOS V4 allows a disk or partition to be larger than 32M. V4 adds a second field in the boot record for disks or partitions with more than 64K sectors. If a disk is formatted with more than 64K sectors, the new sector number field is used and the previous sector number field is set to 0. When DOS V4 builds the disk's FAT, it adjusts the cluster's sector multiple upward as disk capacity increases. By adjusting the cluster, the FAT's size can remain constant while covering the additional sectors of a disk larger than 32M.

Because DOS versions prior to V4 are not designed for extensions that accommodate disks larger than 32M, you cannot expect to find everything you stored on a large capacity disk (larger than 32M) if you boot your PC with an earlier version of DOS. Some third party disk utility applications programs do not allow for DOS V4's extensions to the boot sector, either. If you have DOS V4, before you use a disk optimizer such as the Norton Utilities or a rapid backup program such as FASTBACK, be sure that the application's version expressly handles DOS V4 disks.

Caution: *Before using a disk optimizer or rapid backup program with DOS V4, be sure that the application's version can handle DOS V4 disks.*

Creating the Root Directory

After initializing the FAT, FORMAT creates the disk's root directory in the control area. The root directory is the last of the FORMAT-created control area entries. (DOS V4 places a volume label in the control area as well.)

Cue: Each disk's root directory is fixed in length.

Each disk's root directory is fixed in length. A root directory cannot shrink or grow because of file activity. Different formats of disks may fix the length of the root directory to a different size than other formats. Disk formats with more capacity allot more sectors to a root directory than do formats with lesser capacity.

Each entry in the root directory consists of a 32-byte record describing either a file, a subdirectory, or a disk volume label. Chapter 2 explains the function of a directory as a basic DOS "bookkeeping" tool. Table 5.2 lists the number of entries available with different disk formats.

Table 5.2
Capacity of Root Directory Entries

Disk Format	*Number of Entries*
5 1/4-inch Floppy Disk	
180K	64
360K	112
1.2M	224
3 1/2-inch Floppy Disk	
720K	112
1.44M	224
Typical Hard Disk	512

The control area is normally located in track 0 of cylinder 0 in any disk or partition or logical disk. Track 0 cannot contain any bad sectors; FORMAT writes, in each sector used, the contents of the control area using a byte-offset method. DOS needs to extract the control area's bookkeeping information from expected points. If FORMAT "skipped around" a bad sector, DOS would not find the necessary information in the control area.

When FORMAT finds a bad track 0, it displays a message similar to the following:

```
Invalid media or track 0 bad-disk unusable
```

You can take one of several courses of action when FORMAT displays such a message. If the disk is a floppy, you can either try to reformat or make sure that the disk is right for the drive capacity. If the disk is a logical drive or partition,

you can delete the partition and move its boundaries over a track or cylinder. You will then have a new logical track 0.

You will see these alternatives in later sections of this chapter. For now, just remember that the control area of a disk or a disk partition must be free of bad sectors.

Reminder:
The control area of a disk or disk partition must be free of bad sectors.

Formatting the Files Area

FORMAT writes sector identification numbers, filler data, and cyclic-redundancy check data in every sector space on a disk. The sector identification is used by DOS when the disk is returning a sector of data after a disk read to ensure that the correct sector was read by the drive. Without this test, the drive can return the wrong sector and the requesting program can lock up or cause a disk crash.

The cyclic-redundancy check, or *CRC*, is a computational result that is derived by a calculation. The calculation uses the sector's byte values to produce a CRC for each sector. If one or more of the sector's byte values is incorrectly written to the disk, a subsequent CRC calculation will not produce the same CRC as the original CRC stored in the sector. A non-matching CRC for the "same" data means that the data contains an error. Disk driver routines use the CRC to ensure data integrity at the sector level.

FORMAT writes filler data into each sector after the sector identifier. The filler data acts as a place holder for the sector's eventual file data. The filler data is immediately re-read by FORMAT to test that sector's ability to faithfully record data. FORMAT considers a sector *bad* (unusable) if the sector area on the disk has a flaw and the filler data is not properly written. If FORMAT finds a bad sector in the files area of the disk, the sector's cluster is marked as bad in the FAT. DOS will not allocate the bad cluster. Remember that if the bad sector is in the reserved control area of the disk (usually track 0), FORMAT must reject the disk as being unusable. The reserved area of the disk contains data whose position is fixed by FORMAT.

Writing a Volume Label

If you use the /V switch with FORMAT, you will be prompted for a volume label (the /V switch is built into FORMAT V4). Prior to V4, a volume label refers simply to a name stored in a special root directory entry. The volume label reduces the total available root directory entries by one. In V4, the volume label is stored in the boot record along with a volume serial number.

Reminder:
The volume label reduces the total available root directory entries by one.

You can use the VOL command to identify a disk by its volume label. V3.2 through V4 require you to give the volume label of a hard disk that you are trying to reformat if the disk currently has a label. FORMAT V3.2 and later versions will prompt you to enter the disk's label as a safety precaution. The DIR

command gives the volume label of the disk specified (or default) in the command. Volume labels do not serve, however, as positive identification because you can freely give two disks the same volume label.

Reminder: *All DOS V4 disks are given serial numbers by FORMAT.*

For identifying V4 disks, DOS has implemented an additional identifier besides the volume label. The DOS V4 serial number is positive identification because it is generated by FORMAT and is based on the time and date the disk is formatted. The SHARE command in V4 checks the serial number of a disk before accessing an open file to make sure that no disk swap has occurred. All V4 disks are given serial numbers by FORMAT.

Copying the System Files

Cue: *The command* ***FORMAT /S*** *copies the DOS hidden system files and COMMAND.COM.*

The command **FORMAT /S** copies the DOS hidden *system* files along with COMMAND.COM from the default disk and directory, or the disk and directory named in the CONFIG.SYS COMSPEC directive, to the disk being formatted. The capacity of the newly formatted disk is reduced by the size of these files, and the disk is bootable. If you know that you want to make a disk bootable, use the /S switch with FORMAT rather than use the SYS command at a later time.

You can be sure that the disk will have room for the system files if you use **FORMAT /S**. SYS will not transfer the system if you have copied any files to the newly formatted disk.

Understanding the General Rules of FORMAT

- FORMAT treats floppy disk drives as physical drives and formats their disks completely.
- FORMAT respects hard disk partitions and their logical drive divisions and formats only one partition or logical drive at a time.
- Formatting a disk can also mean formatting a logical disk partition.
- A hard disk must be partitioned before you can format it even if the entire disk is used as one partition.

  If your hard disk is divided into drives C:, D:, and E: by partitioning, using the FORMAT command on drive E: will not affect drive C: or D:, for example.
- You cannot use FORMAT to format from your PC a drive which is on another PC in a network recognized by DOS.

  Most networks use or emulate DOS network service functions. In fact, you must be logged off a network to format a disk on your PC unless

you "pause" your network session. (Check your specific network operating system's manual to confirm your limitations when using FORMAT.)

- FORMAT cannot prepare a disk which is being rerouted by an ASSIGN or SUBST command.
- You cannot format a VDISK or other RAM disks unless your version of DOS specifically supports a specialized RAM disk.

  For example, some Toshiba laptops have a provision for a configurable "hard RAM" disk which must be formatted before it can be used.

  NOTE: The Toshiba hard RAM disk is a RAM disk that uses expanded memory. Unlike a RAM disk created by VDISK, the Toshiba hard RAM disk does not lose its contents when the laptop computer is turned off. As long as the battery has power, the contents of hard RAM stay intact. Consult your Toshiba operations manual if you have a Toshiba PC and use a hard RAM disk.

Formatting a Diskette

As an example of using the FORMAT command, you can format a bootable diskette. If you are comfortable with formatting diskettes, you can skim this exercise. Be sure not to include the drive letter of your hard disk in the command line; you do not want to format your hard disk by mistake. Because FORMAT is an external command, you must include its directory in your PATH, make its directory your current directory, or log to drive A: and use your working DOS disk. As with other external DOS commands, you can include the command's path name as part of the full command and therefore directly specify to DOS where the FORMAT command is located.

Type the following command line and press Enter. (DOS V4 users can omit the /V; the switch is automatic in V4.)

FORMAT A: /S/V

/S and /V are the optional switches available with FORMAT. Remember that to use a switch, you type a slash (/) and the character for the switch. A switch is always typed on the same line as that of the DOS command and is usually the last item typed on the line. Some switches require extra information. FORMAT's switch options are listed in table 5.1.

After you type the FORMAT command line, the red light on the disk drive may go on briefly. You will see the following message:

```
Insert the diskette for drive A:
and strike ENTER when ready
```

Reminder:
To abort the operation of the FORMAT command, press Ctrl-C instead of Enter.

If you are using a working DOS disk, take it out of drive A:. Place a blank or reclaimed disk (a disk containing no useful files) in drive A: and press Enter. The FORMAT command will not continue until you press Enter. You can abort the command at the prompt by pressing Ctrl-C instead of Enter. FORMAT is one of the rare commands for which DOS wants you to press a *specific* key, not just *any* key, before it carries out the command. When you are formatting disks, you do not want DOS to let a stray coat sleeve against the space bar send your working DOS diskette or some other important disk into a new format.

After you press Enter, the red light on the first disk drive will go on again, and the following message will appear if you have DOS V3:

```
Head:   0 Cylinder:    1
```

If you have DOS V4, you will see this:

```
1 percent of disk formatted
```

The numbers on-screen will change constantly. For DOS V3, the number after the word `Head` flips between 0 and 1. The number after the word `Cylinder` starts at 1 and increases to either 40 (for 360K minifloppies) or 80 (for HC minifloppy disks and for microfloppy disks).

DOS V4 gives you a continual update of what percentage of the disk has been formatted. After 20 to 40 seconds, the line with the numbers disappears, and the following messages appear in its place:

```
Format complete
```

After a few more seconds, you will see another message:

```
System transferred
```

Then DOS asks the following:

```
Volume label (11 characters, ENTER for none)?
```

Type **DOS_Disk_01** and press Enter.

Alternatively, you can type **DOS DISK 01** and press Enter. Spaces are allowable in volume labels.

The diskette will spin, and DOS will tell you how much total information the disk can hold, how much space is taken up by DOS, and how much free space is on the disk. For DOS V3.3, the figures are as follows:

Disk Size	*Message*

360K	`362496 bytes total disk space`
	`78848  bytes used by system`
	`283648 bytes available on disk`

```
1.2M        1213952 bytes total disk space
            78336  bytes used by system
            1135616 bytes available on disk

720K        730112 bytes total disk space
            78848 bytes  used by system
            651264 bytes available on disk

1.44M       1457664 bytes total disk space
            78336 bytes  used by system
            1379328 bytes available on disk
```

Remember that the numbers for various versions of DOS V3 will be slightly different.

DOS V4 users will see slightly different results, such as the following results from a 1.44M disk:

```
1457664 bytes total disk space
107520 bytes used by system
1350144 bytes available on disk
512 bytes in each allocation unit
2637 allocation units available on disk
Volume Serial Number is 3952-0BE4
```

Note that DOS V4 reports the number of bytes taken up by each allocation unit, which is the cluster size in bytes. By dividing the number of bytes in each allocation unit (cluster) by 512 (the sector size in bytes), you can determine the number of sectors in a cluster for the disk type being formatted. This screen report also tells you the number of allocation units or clusters available on the disk. Remember that these information items are part of DOS's internal bookkeeping tables for the disk's file system.

In addition, DOS V4 assigns to each disk a unique serial number, which it calculates from an internal algorithm. As you learned in a previous section, this unique serial number enables DOS to keep track of disk changes, even if many disks have the same volume number or name.

You may see one of the following messages on-screen:

```
1024 bytes in bad sectors
```

or

```
Invalid media or track 0 bad--disk unusable
```

The first message means that DOS found on the disk some bad sectors that cannot be used to hold information. Your disk is usable, but the total amount of free space on the disk is reduced by the number of bytes in bad sectors.

The second message means that the areas on the disk that hold key DOS information are bad, and the disk cannot be used at all. Chances are good that you won't see one of these messages. If, however, one of these messages appears, answer Y to the following question to try the process again with the same disk:

```
Format another (Y/N)?
```

Reminder: *Some diskettes that do not format correctly the first time do so the second time.*

Press Y and then press Enter. You will see a repetition of the messages that instruct you to insert a floppy disk to be formatted. You'll be using the same disk that showed problems. Don't press Enter at the prompt yet. First, take the disk out of the drive, reinsert the disk, and close the door if necessary. The disk may not have been seated correctly in the drive the first time, or a piece of dust may have gotten between the disk and one of the drive's heads. Some diskettes that do not format correctly the first time do so the second time. Now press Enter to confirm that you are ready; the format process will repeat.

If the disk does not format properly the second time, something may be wrong with either the disk drive or the disk (most likely the disk). You can take a guaranteed disk that has some bad sectors back to the dealer for a replacement, or you can use it, knowing that you will have less storage space than you should have. If the disk is old, it may be starting to deteriorate and FORMAT has detected the bad sectors. Throw the old disk away.

If you use a deteriorating disk, files stored on the disk may become corrupt; you will then not be able to use the files.

The message `Invalid media` can appear if you use a 360K disk in an HC disk drive or a 720K disk in a 1.44M disk drive. If you used the wrong type of disk, get the right type and try this exercise again. HC and 1.44M disks have special magnetic coatings designed to work with higher data densities. (Data densities and drive designs are discussed in Chapter 2.) If you used the right type of disk and got this message, take the disk back to your dealer for a replacement. If the message `Invalid media` often appears while you are using valid media, you may have a faulty drive. Two or three failed disks per box of ten is excessive when you are buying good quality disks. The problem may be with your hardware.

FORMAT should now be asking the following question:

```
Format another (Y/N)?
```

Caution: *Never use a ball point pen to write on the label after it is attached to a 5 1/4-inch diskette.*

Answer N and press Enter. The DOS prompt should reappear. Even if you have formatted bootable diskettes before, use this exercise to help you look closely at the process. As always, you should label the newly formatted disk, using a label specially designed to adhere to diskettes. A post-it note has a life expectancy of 30 seconds when used as a disk label. The same post-it note, when dislodged in your drive, can jam up your drive's mechanical parts intermittently for years. Remember also never to use a ball point pen to write on the label after it is attached to a 5 1/4-inch diskette. Use a felt-tip pen only.

Formatting a Diskette with the DOS Shell

Using the DOS Shell to format a bootable diskette is much like formatting from the DOS command line. The main difference is that you access the FORMAT program from the Shell instead of directly from the command line. The DOS Shell requires you to type the command line parameters just as you did in the preceding exercise (see fig. 5.1).

Cue: *DOS V4 users can access the FORMAT command from the DOS Shell instead of directly from the command line.*

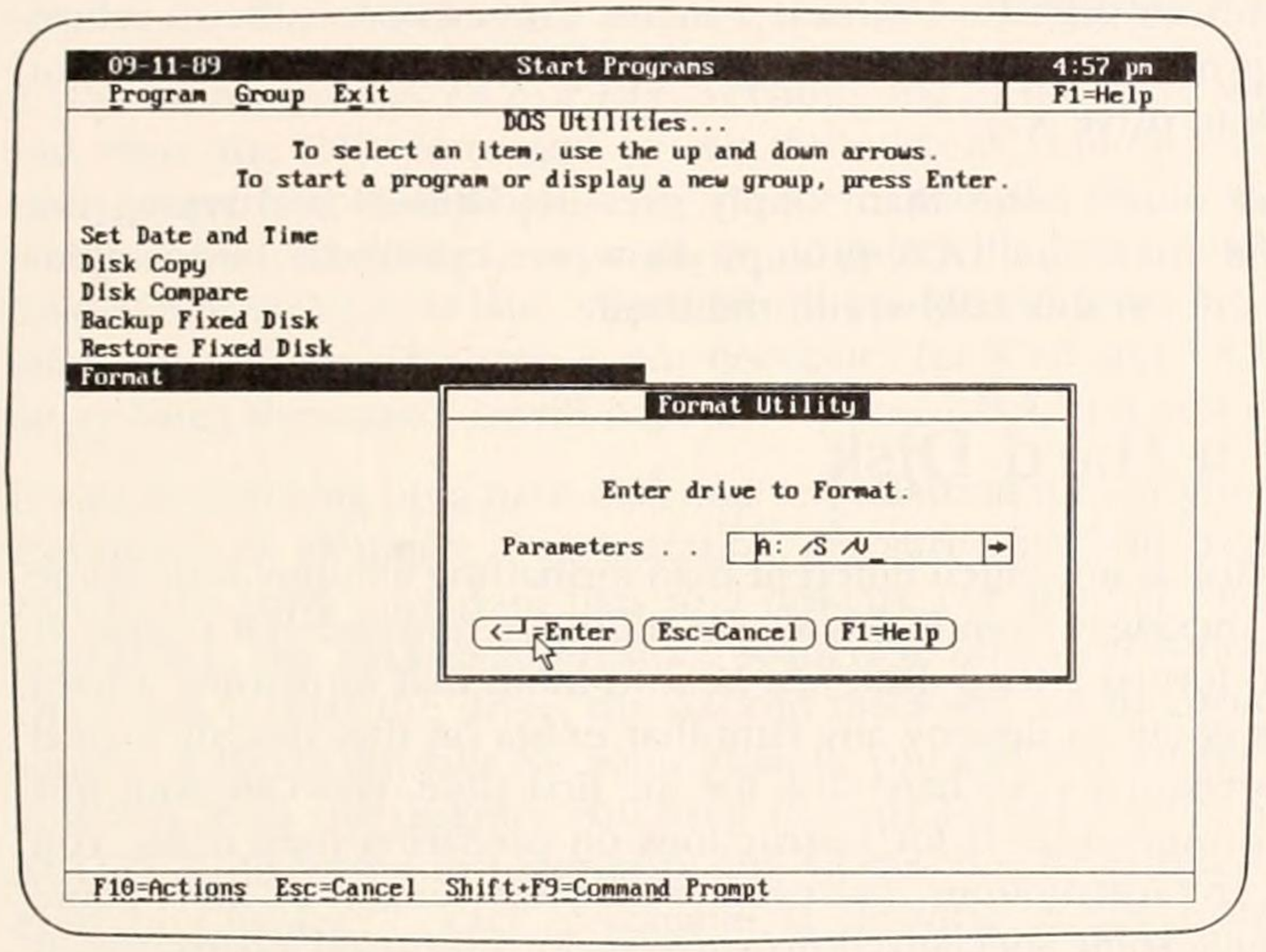


Fig. 5.1

The Format Utility screen from the Start Programs menu in the DOS Shell.

To format a floppy disk, follow these steps:

Step 1. *Insert into drive A: a fresh disk to be formatted.*

Step 2. *Use the cursor keys or mouse to highlight the selection DOS Utilities. . . from the Main Group menu. Press Enter or click the mouse button.*

If DOS Utilities. . . is highlighted, you can also access this menu by choosing Start from the Program menu in the Start Programs action bar. If your computer doesn't have a hard disk, you must leave the DOS Shell disk in drive A: when selecting utilities.

Step 3. *Select the Format option.*

DOS shows you a command line labeled `Parameters`. Because FORMAT is most often used to prepare a floppy disk, DOS supplies the drive name A: as a default value at the beginning of the parameter list. In this case, however, you want to add the /S and /V switches.

Step 4. *Type the switches /S and /V.*

You do not need to type the FORMAT command itself.

Step 5. *Press Enter or click the Enter box.*

To terminate the FORMAT program, press Esc or click the Escape box by using the mouse cursor.

The menus will be erased from your screen, and you will see much the same screen display as when you format a disk from the DOS prompt. When DOS is finished, it will return to the DOS Utilities. . . menu. You can press Esc to return to the Main Group menu. (If you are using floppy disks, you will have to put the Shell disk back in drive A:.)

This method is not much faster than simply pressing Shift-F9 and typing the FORMAT command from the DOS prompt. However, you will find helpful knowing how to perform this task within the Shell.

Formatting a Hard Disk

Reminder: *Appendix D contains complete instructions for preparing and formatting hard disks.*

Formatting a hard disk is not much different than formatting a floppy disk. Some of the preliminary messages from FORMAT are different, however. Of course, it takes more time to format a hard disk. Just bear in mind that formatting a hard disk or partition is going to destroy any data that exists on that disk or logical drive. If you are preparing your hard disk for the first time, you can skim this section and refer to Appendix D for instructions on preparing hard disks. You should also check any instructions that came with your version of DOS. Some manufacturers include some specialized instructions for preparing a hard disk. If you have any doubts about the process, consult your computer dealer before you format your hard disk.

Completing Preliminary Steps

In order to format a hard disk, you should complete some preliminary steps that will make your hard disk formatting experience more predictable. Because you will not be formatting your hard disk often, refer to this section each time you format a hard disk.

Before you can format a new hard disk, you must partition it. You format separately the primary DOS partition and any logical drives of an extended DOS partition separately. Some hard disks are partitioned by the dealer before users get them. If you have any doubt about your hard disk's partition status, run FDISK from your floppy drive to check for partitions. See the section on FDISK in this chapter; also Appendix D explains in detail the operation of FDISK.

Always have a working copy of the master DOS disks or the DOS V4 working disks available before you format the hard disk. If you format the active DOS partition, chances are good that you will be destroying the external DOS commands that reside on your hard disk. The directory that holds the external DOS commands (usually \DOS or \BIN) will not be there when the format is completed. Without the DOS working copy, you will not have access to the external DOS commands. Also make sure that your floppy drive is in reliable condition before you begin the format. Run CHKDSK on your working copy of the DOS disk(s) to ensure that the disk is all right. (For instructions on using CHKDSK, see the Command Reference and the section on CHKDSK in this chapter.)

Caution:
Always have available a working copy of the master DOS disks before you use FORMAT on a hard disk.

If you are using DOS V3.2 or later versions, log to the drive you want to format and issue the DIR command to see the current volume label. If you cannot easily remember the label, write it down with its exact spelling (including underscores and spaces) on a piece of paper. Before actually formatting the disk, FORMAT V3.2 or later versions will ask for this label for confirmation as a safety precaution. This step is not necessary for V3.0 and V3.1. If the drive has never been formatted, it will have no volume label, and you can skip this step.

If you are backing up a hard disk and reformatting it before using the RESTORE command to eliminate fragmentation, be sure that you have used COPY or XCOPY to copy important files to a diskette (or another logical drive of your hard disk). The backup-and-restore sequence, however, is not 100% foolproof. Once you format the drive, the backup disks will be the primary file-recovery source. If RESTORE fails for some reason, you can at least recover your important files from the diskette you used to copy them to with COPY and XCOPY. XCOPY is discussed in Chapter 6, and COPY, BACKUP, and RESTORE are covered in Chapter 7. Each command is detailed in the Command Reference section.

Even if you are reformatting a drive that contains data you do not think you want to keep, there is still a good chance that you may need in the future at least one or two files in one of that disk's directories. Because all files and directories on the disk you intend to format will be destroyed, you may want to take a final look at the file names in the root and subdirectories to make sure that an important file isn't buried somewhere. One handy command to use for this last view is the TREE command. (A complete discussion of TREE is found in the Command Reference section; the general discussion of TREE is found in Chapter 6.)

Cue:
Issue the TREE command on the disk you want to format to make sure that you won't be destroying an important file.

Use the following command:

TREE *d:*/**F**

Substitute the letter of the drive you want to format for *d:*. This command with the /F switch produces a listing of the directory structure with each directory's file names. For a printout of the output, use the following command:

TREE *d:*/**F** >**PRN**.

Again, substitute your drive name for *d:*. This command redirects the TREE output to your printer. Use the output to assure yourself that formatting the disk will not destroy anything you want to keep.

Issuing FORMAT for the Hard Disk

Assume that the hard disk (or logical disk) that you are going to format is drive C:. If you are formatting another drive, use its drive name (letter) in place of C. Note that if your drive is not the primary DOS partition, you do not need the /S switch because DOS will boot from the primary DOS partition only. Put your working copy of the DOS master system disk in drive A:, log to that drive by typing **A:**, and then type the following:

FORMAT C:/S/V

If the drive has already been formatted, it has a volume label; and you are using V3.2 or later versions, FORMAT will display the following prompt:

```
Enter current Volume label for drive C:
```

Enter the current volume label for the drive. If the volume label does not match what FORMAT found on the disk, you will see the following message:

```
Invalid Volume ID
Format failure
```

If this message is displayed, make sure that you are formatting the correct drive. Check the volume label again. If the label appears to be correct, it may include a hidden character. You can change the volume label of the disk to one of your choosing with the LABEL command (discussed later in this chapter) and use the new volume label to successfully answer the volume label confirmation prompt.

FORMAT will issue the following warning message and confirmation prompt:

```
WARNING, ALL DATA ON NON-REMOVABLE DISK DRIVE C:
WILL BE LOST!
Proceed with Format (Y/N)?
```

Cue:
The time it takes for the FORMAT process to be complete varies depending on the capacity of the disk being formatted.

Examine this prompt to confirm the disk drive name (letter) before you answer Y. If you get in the habit of answering Y to the confirmation prompts of less dangerous commands, you may make a serious mistake with this final FORMAT confirmation prompt. If the drive is the one you intend to format, enter Y. If not, enter N to terminate FORMAT.

FORMAT will then update the display with progress reports on its initialization activity. V4, as in the preceding example on formatting floppy disks, reports percentage completion. Other versions report the head and cylinder count. The process can take from a few minutes to more than a half-hour; the length of

time is related to the capacity of the disk being formatted. Larger disks take more time. When the disk has been formatted, FORMAT issues the following message:

```
Format complete
```

FORMAT may not be finished, however. If you used the /S switch, you will see the following message:

```
System Transferred
```

If you used the /V switch, or if you omitted the /V switch using V4, FORMAT prompts for the volume label with the following:

```
Volume label (11 characters, ENTER for none)?
```

Enter the label text you want. The discussion of LABEL in this chapter includes a list of allowable volume label characters. If you need to change this label, you can use the LABEL command in V3.1 (PC DOS V3.0) and later versions.

Finally, FORMAT will display a report showing the disk space formatted, the bytes used by the system files, defective sectors marked (if any), and the number of bytes available on the disk. Don't be surprised if your hard disk shows some bad sectors in the report. It is not uncommon for hard disks to have a few bad sectors, especially if the formatted capacity exceeds 21M. These bad sectors are marked in the FAT and will not be allocated to a file.

On the other hand, if you purchase hard disk packs, you can specify "error-free" for no bad sectors, or "flag-free" to mark questionable sectors. New disk media is more reliable than that of five years ago. You can only tell if the disk has bad blocks by checking the defect table that is glued to the drive. If the disk controller supports bad-sector mapping, the drive gets an alternate sector from an included alternate track. DOS never sees the switch.

Naming Disks with LABEL

The external command LABEL adds, modifies, or changes the volume label of a disk. In DOS, a volume label is a character name given to a physical or logical disk (the volume). The LABEL command is available with PC DOS V3.0 and later versions and MS-DOS V3.1 and later versions. If a disk's volume label is blank (if the user pressed Enter when FORMAT or LABEL prompted for the label), you can use the LABEL command to add a volume label. Volume labels are passive. They do not affect any DOS operation. Labels merely show the identity that the disk was given during formatting or the last entry of the LABEL command.

Cue:
Give each disk (physical and logical) a volume label so that you can more easily identify the disk later.

DOS displays the volume label when you issue commands such as VOL, CHKDSK, DIR, and TREE. DOS V3.2 FORMAT asks for a hard disk's volume label

before reformatting the disk. It is a good idea to give each disk (physical and logical) a volume label. You can more easily identify the disk when it has a unique volume label.

Issuing the LABEL Command

The syntax for the LABEL command is as follows:

LABEL *d:volume_label*

d: is the drive name that holds the disk to be labeled. *volume_label*, the optional label text that you supply as the new volume label, can include from 1 to 11 characters. If you do not supply the *volume_label* parameter, LABEL will automatically prompt for a new label.

Understanding the Operation of LABEL

Unlike FDISK and FORMAT, the external command LABEL does not affect user directories or files. When you issue a LABEL command alone or with an optional drive specifier, DOS accesses the special file entry in the root directory of the default or specified disk and finds the volume label characters in the file-name field.

Cue: *The LABEL command can accept up to 11 characters of volume-label text.*

Because DOS directories provide 11 characters for a file name (8 for the name plus 3 for the extension) LABEL can accept 11 characters of volume-label text. DOS displays these characters as the volume label or indicates that the volume (disk) has no label if the name is blank. DOS V4 also stores the volume label in an extended field of the boot record in the disk's control area where it can be speedily accessed without a directory search. DOS then prompts you for a new label of up to 11 characters. You can enter the new label at this prompt or press Enter for none. If a label is entered, DOS updates the volume label entry in the directory with the new label. If no label is entered (the Enter key is pressed alone), you are prompted to indicate if you want the current label deleted. Answer Y to delete the current label or N to leave the current label intact.

The command sequence looks like this:

LABEL

DOS responds:

```
Volume label in C is BOOT DISK
Volume label (11 characters, ENTER for none)?
```

If you press Enter in response to the prompt, DOS confirms your decision by prompting as follows:

```
Delete current volume label (Y/N)?
```

When you press Y and then Enter, DOS deletes the current label. If you press only Enter or with any character other than Y or N, DOS repeats the prompt.

When you issue a LABEL command with the optional *label* parameter of from 1 to 11 characters, DOS immediately updates the specified or default drive's label with no warning prompt. If you enter an incorrect volume label, simply enter the LABEL command for the drive again and correct the label.

Caution: *When you issue a LABEL command with an optional parameter of from 1 to 11 characters, DOS updates the specified or default drive's label with no warning prompt.*

Understanding the General Rules of LABEL

- The LABEL command will only work on a disk after the disk has been formatted.
- Volume labels can include from 1 to 11 characters.
- The label can contain any alphabetic (upper- and lowercase) character (letter) or any numeric character. DOS will translate lowercase letters into uppercase letters before storing them in the label.
- The following special characters and punctuation symbols are allowed:

  `$ # @ ! ( ) { } ' _ ~`

- The extended (ASCII 128 and greater) characters produced by holding down the Alt key and then selecting a number from the numeric keypad are allowed.
- The space character is allowed, but for V3.2 and earlier versions, you must use the LABEL command with no label parameter so that you can enter the label when prompted.

  DOS will not interpret the space as a part of the label parameter in the command line as you intend it to. In the command **LABEL A:MY FLOPPY**, for example, DOS sees the space between Y and F as a command-line delimiter and concludes that the label is incorrectly formed. The same label entered at the prompt `Volume label (11 characters, ENTER for none)?` will be accepted with no objections.
- The following characters are *not* allowed with the LABEL command:

  `* ? / \ | . , ; : + = < > [ ]` Esc Tab
- Any control characters (ASCII 31 and lower) and the DEL character (ASCII 127) are not allowed.

- You cannot use the LABEL command on a networked drive.
- With V3.2 and later versions, you cannot use LABEL on a disk in a drive that is affected by the SUBST or ASSIGN commands.

Labeling a Disk

If you formatted without a volume label a diskette that you intend to use as a work disk without a volume label, you can place a volume label on the disk by placing it into drive A: and issuing the following command:

LABEL A:WORK_DISK1

The disk in drive A: will immediately have the WORK_DISK1 volume label.

If you have a diskette with an existing label and want to identify it as a work disk, place the disk in drive A: and issue the following command:

LABEL A:

Then follow the prompts:

```
Volume in drive A: is SAMPLE
Volume label (11 characters, ENTER for none)?
WORK_DISK2
```

The volume label of the diskette in drive A: will now be WORK_DISK2.

If you want to delete an existing volume label from your hard disk (C:), enter the following command:

LABEL C:

Then follow the prompts:

```
Volume in drive C: is HARDDRIVE
Volume label (11 characters or Enter for none)?
<Enter>
Delete current volume label (Y/N)?
```

Press Y and then Enter; the volume label of drive C: will be deleted.

Some third party programs can edit the volume label of a pre-V3.1 (PC DOS V3.0) formatted disk in a manner similar to LABEL. Check with your dealer for a recommendation of one of these third party programs.

To understand the full benefit of the LABEL command, you should also review the FORMAT command, which is covered in the preceding section. The VOL command is discussed in the next section.

Examining Volume Labels with VOL

The internal command VOL is a convenient command for seeing a disk's volume label or verifying that a label exists. VOL is a display-only command. It will not modify the current volume label. VOL will show the volume label created during the disk's formatting or modified by a subsequent LABEL command.

Cue:
The VOL command displays the volume label created during the disk's formatting or modified by a subsequent LABEL command.

VOL accesses the disk's volume label from the root directory and displays the label. VOL does not change any files or the label name, so you can use it freely. The command accommodates an optional drive specifier and uses the default drive if no drive specifier is given.

Issuing the VOL Command

To see the volume label of the disk in drive A:, enter the following command:

VOL A:

DOS will respond with the following message:

```
Volume in drive A: is WORK_DISK1
```

CHKDSK, TREE, and DIR include the volume-label display as part of their outputs.

Understanding the General Rules of VOL

- If you do not specify a drive parameter, the current drive is assumed.
- A volume label is added to a disk during formatting with the /V switch or after formatting using the LABEL command.
- The VOL command displays the same volume label for a disk as the DIR, CHKDSK, and TREE commands.

Transferring the DOS System Using SYS

All DOS disks have a DOS file system, but only disks with the DOS system files and COMMAND.COM can be used to boot DOS. The external command SYS transfers (copies) the hidden system files from a bootable system disk. You then must use COPY to transfer COMMAND.COM to the target disk, if desired, because SYS does not transfer COMMAND.COM.

Cue:
The external command SYS transfers (copies) the hidden system files from a bootable system disk.

Issuing the SYS Command

The syntax for the SYS command is as follows:

SYS d:

If you are using DOS V4, you can also provide the source drive for the system files as in the following:

SYS d: d2:

The drive specified by **d:** is the target drive for the system files. You must include a drive name (letter) for drive **d:**; the drive specified by **d2:** (V4 only) is the drive that is used for the source of the system files.

Caution: *The syntax of the SYS command follows an order of to-source-from-target; this order is the reverse of most DOS commands.*

Notice that the drive specifiers in the V4 syntax example are given in an order that is reversed from the order of most source to target drive specifiers. The normal DOS order, as in **DISKCOPY A: B:**, has data flowing in a from-source-to-target syntax presentation. SYS has a syntax presentation of to-source-from-target.

Understanding the Operation of SYS

SYS can make bootable a disk that was not formatted with the /S switch. In addition, SYS can transfer a new version of the operating system to existing bootable disks. There is, however, no guarantee that SYS will be able to transfer the system files to a disk. The disk must be able to accommodate the system files SYS intends to transfer, and those system files must be compatible with the version of operating system the PC is currently running.

As you recall from Chapter 3, DOS loads the two system files from the boot disk into memory where they act as the default system drivers and the operating system kernel. You do not see these files when using a DIR command because they are hidden. The two (sometimes three) system files are also marked with a system attribute and read-only attribute.

SYS acts as a modified version of the COPY command and can both read the hidden files from the source disk and also bypass the read-only status of existing system files on the destination disk. No other DOS commands can manipulate the system files in this manner. SYS V3.3 or earlier versions will read the system files from the default disk for the transfer even if the SYS command path is given for another drive. DOS V4 SYS allows an optional source drive for the system files. SYS never requires a path to the source system files because these files, if they exist, will always be in the default or V4 source disk's root directory. DOS will not SYS a disk to itself.

Understanding the General Rules of SYS

- SYS requires the destination disk to be formatted.
- The destination disk must be empty (containing no files or directories) or the disk must already contain the two system files.
- With versions prior to V4, if the destination disk contains the two system files, the system files that SYS tries to transfer to the destination disk must not be too large to replace the existing system files. SYS V4 can transfer the system files as long as there are two root directory entries available on the disk you are putting the system on.
- With V3.3 or earlier versions, if a disk already contains user files and does not contain the two system files, DOS will issue the error message `No room for system on destination disk`.

  If the existing system files are too small to allow a new version's system files to take their space, DOS will issue the error message `Incompatible system size`.
- A disk formatted with the /B option can be the destination disk for SYS even if the disk is not empty because the /B switch reserves room on the disk for the system files.

  The /B FORMAT switch forces the disk to an eight-sector-per-track format which is compatible with V1 disk formats. Because an eight-sector-per-track format is normally not desirable, you should not use the /B switch with FORMAT simply to reserve space on a disk for the system files should you decide to store files on the disk and subsequently use SYS on the disk.
- You cannot use SYS on a networked drive. If you want to use SYS on a networked drive, you must either log off the network or pause your drive to perform the SYS command. For exact restrictions, consult your system's network documentation.
- You must give the destination drive parameter for the SYS command. SYS is a command that does not use the default drive for the destination parameter. The source drive for the system files is always the default drive unless you specify a source drive parameter when using SYS V4.
- Although it is possible to use SYS to transfer the system files between some different versions and implementations of DOS, some versions or implementations are not compatible with certain others. If DOS detects incompatibility, the system files will not transfer, and the error message `Incorrect DOS version` will appear.

Caution:
Do not mix DOS versions or implementations when you use the SYS command.

- If you are in doubt about the version of DOS on the boot disk, you can issue the VER command; DOS will display the version. Remember that one of the hidden files is the input-output (IO) system which contains device drivers for the implementation of DOS on a particular computer. Although many implementations of the IO system file are compatible with one another, some may not be. For this reason, you should not mix DOS versions or implementations when using the SYS command.

Using SYS To Transfer DOS to a Diskette

Assume that you are using DOS V4 with a hard disk and have an empty diskette that you want to make bootable to use with another computer. Put the formatted diskette in drive A: and be sure that your current PATH setting includes the directory that contains SYS. Then, enter the following command:

SYS A: C:

The C: is optional; if you are logged to drive C:, C: is already the default source for the system files. SYS will reply with the following message:

```
System transferred
```

The system files are now on the disk in drive A:. To complete the process of making the disk in drive A: a bootable disk, copy COMMAND.COM from the root directory (on some systems, from the DOS subdirectory) of C: to the disk in A: with the following command:

COPY C:\COMMAND.COM A:

The disk in drive A: is now a bootable disk. SYS transferred the system files, and COPY transferred COMMAND.COM.

Copying Disks with DISKCOPY

Reminder:
You can use DISKCOPY with floppy disks only; DISKCOPY cannot be used to copy the contents of a hard disk to another hard disk.

You can use the external command DISKCOPY to copy an entire disk's contents to another with a like disk format. DISKCOPY works with floppy disks only; you cannot use DISKCOPY on a hard disk. Unlike the file-based COPY command, DISKCOPY is a sector-based command. DISKCOPY reads and writes in sectors rather than in files. The destination (target) disk of a successfully completed DISKCOPY is a mirror image of the source disk. After you issue the DISKCOPY command, DOS waits for you to get the appropriate disks in the drives until you press any key. The disk copying process then begins.

DISKCOPY is a disk-level command because it cannot copy a disk file by file. DISKCOPY replicates each sector of the source disk on the destination or target

disk. Because DISKCOPY copies every sector from the source disk, DOS's boot sector, FAT, and root directory are mirrored on the destination disk. Even the files area of the source disk is replicated on the destination disk by DISKCOPY.

COPY and XCOPY do not copy a source disk's boot sector, FAT, and root directory. COPY and XCOPY technically copy sectors, but COPY and XCOPY use the source disk's directory to copy a file's worth of sectors at a time. DISKCOPY starts at track 0, side 0, sector 1 and copies every sector in order. If the source disk is bootable, DISKCOPY makes a bootable destination disk. DISKCOPY does not know that the system files on a source disk are hidden files. DISKCOPY never consults the source disk's directory. You can use DISKCOPY on a disk that contains no files or 100 files. The DISKCOPY operation is exactly the same regardless of the number of files on a disk.

Reminder:
The DISKCOPY operation is exactly the same regardless of the number of files on a disk.

Issuing the DISKCOPY Command

The syntax for DISKCOPY is as follows:

DISKCOPY *d1: d2: /1*

d1: is the drive that contains the disk you are using for the source of the copy (the original).

d2: is the drive that contains the disk targeted for the resulting copy (the duplicate).

The optional */1* switch specifies that the disk copy will be single-sided rather than double-sided. The /1 switch assures compatibility with V1.0 single-sided disks.

Understanding the Operation of DISKCOPY

Remember that the logical structure of a disk is based on the number of cylinders, heads, and sectors per track (see Chapter 2 for more information on logical structure). The disk drive's low-level driver program translates DOS's logical sectors to a drive's physical cylinder, head, and sector locations. DISKCOPY requests logical sectors from the source disk and writes them in order to the destination disk.

Normally, the type of drive used for the DISKCOPY source and destination disks is the same. You can use DISKCOPY on a 5 1/4-inch disk to copy to a 5 1/4-inch disk or a 3 1/2-inch disk to a 3 1/2-inch disk. The source and destination disks must have the same logical DOS format. DOS consults the disk's boot record to ensure that the formats are the same before the DISKCOPY operation. The key here is the disk's format, not the disk's physical size.

DOS views disk formats as logical formats and does not "see" the physical disks involved in a DISKCOPY. You can use optional FORMAT switches to format a 5 1/4-inch HC disk to a 720K format and format a 1.44M HC 3 1/2-inch disk to 720K. DISKCOPY sees the two disks as being equal because the track and sector per track density for the two different physical sizes is logically the same.

You can then use DISKCOPY to copy the 5 1/4-inch disk to the 3 1/2-inch disk because both disks are 720K disks. Note that DOS V3.3 and later versions enable you to produce 720K formats on both 5 1/4-inch and 3 1/2-inch disks in high capacity drives.

If the destination disk is not formatted, or if its format is not the same as that of the source disk, DISKCOPY will try to produce the source disk's format on the destination disk as a preliminary step in the copying process. DISKCOPY's ability to produce target formats is directly linked to the formats supported by the disk-driver routines.

Reminder: *DISKCOPY cannot produce a destination format that the destination drive (or driver routine) does not support.*

DISKCOPY cannot produce a destination format that the destination drive (or driver routine) does not support. You cannot, for instance, use DISKCOPY to copy a 1.2M disk to a disk in a 360K drive.

DISKCOPY reads as many of the source disk's sectors as available system memory will allow before writing those sectors to the destination disk. In a single floppy disk system, this all sector-read method results in fewer disk swaps of the source and destination disks than the command **COPY** *.* would require to complete the DISKCOPY. COPY will read and write one file at the most before requiring a disk swap on a one-floppy system. In a system with 640K of available memory, DISKCOPY will read an entire 360K floppy into memory in one read operation before it writes the source disk's contents to the destination disk. In a system with 256K of available memory and one floppy drive, you will have to swap the source and destination disks into the drive during the course of the disk copy.

Reminder: *DISKCOPY does not eliminate file fragmentation.*

Because DISKCOPY reads sectors—not files—and writes the target disk in sector units, the source disk's directories, FAT, volume label, and bad sectors (if any) are reproduced on the target disk. DISKCOPY transfers, by way of sectors, any hidden and system files that may be on the source disk to the target disk. Files on the destination disk are fragmented to the same extent as the files on the source disk. DISKCOPY does not eliminate file fragmentation. If you want to eliminate fragmentation, use the COPY command to copy files to a freshly formatted disk. (Fragmentation is discussed in Chapter 2; COPY is discussed in Chapter 7.)

Any files present on the target disk before a DISKCOPY are overwritten by DISKCOPY. You cannot add files to existing files using the DISKCOPY command, so use the COPY command to add files to existing files on a disk. Like FORMAT, DISKCOPY destroys the contents of the target disk. Always write-

protect the source disk! It is very easy to get the source and destination disks reversed when using DISKCOPY.

Caution: *Always write-protect the source disk when using DISKCOPY; like FORMAT, DISKCOPY destroys the contents of the target disk.*

Understanding the General Rules of DISKCOPY

- DISKCOPY works only with floppy disks. DISKCOPY cannot copy to or from hard disks.
- If the target disk is formatted with a format different from that of the source disk, DISKCOPY will format the target with the same format as the source.
- If you do not supply source and target parameters, DISKCOPY will use the current drive for both the source disk and also the target disk. You may have to swap the disks during the copy process more than once.
- If the target disk contains bad sectors, DISKCOPY will abort and display a message indicating that the target disk is unusable.

  Because the source disk's FAT is duplicated on the destination, any sectors marked as bad on the source disk will be marked as bad on the destination disk even if the destination disk has no "truly" bad sectors.
- You should format target disks with FORMAT before using them with DISKCOPY if you want to see the bad sector report produced by the FORMAT command. When DISKCOPY formats a target disk, it produces no formatting summary report.

  For more information on the report issued by FORMAT, see the section on the FORMAT command in this chapter.

Using DISKCOPY To Copy the DOS Startup Disk

Several times in this book, you will be reminded to have a working copy of your master DOS disks. In this example of using the DISKCOPY command, you will make a copy of the original disk or disks that hold DOS. If you do not have a working copy of your DOS disks, try this example.

If you have a microfloppy (3 1/2-inch) disk drive, you will make a copy of the single microfloppy disk that comes with DOS. If you use a minifloppy (5 1/4-inch) disk drive, you will copy the minifloppy disks that come with DOS (one

or two with DOS V3.3 and earlier versions; five with DOS V4). If you have other disks in your DOS package, you can use this example as a model to copy them also.

First, you will make a copy of your DOS V3.3 (or earlier versions) or DOS V4 Startup disk. This is the disk you used to load DOS. With DOS V2 and V3, you can run DOS from a floppy disk using this disk only. For DOS V4, you will be able to load DOS from the copy of the disk you will make, but you also have an additional option. You can create a start-up disk with the V4 SELECT program. Do not use the V3.3 or earlier SELECT program for making working copies of DOS; V3.3 and earlier versions of SELECT contain a provision to format your hard disk.

Caution:
Do not use SELECT V3.3 or earlier versions to make working copies of DOS; these earlier versions can format your hard disk.

The disks you will be copying are those shown in table 5.3.

Table 5.3
DOS Disks for Versions 3 and 4

DOS Version	*3 1/2-Inch Disks*	*5 1/4-Inch Disks*
V3	Startup/Operating	Startup/Operating (or may be called Supplemental Programs)
V4	Install Operating	Install Select Operating 1 Operating 2 Operating 3

This exercise includes two sets of instructions. Follow the first set if you have a computer with two floppy disk drives; follow the second set if you have one floppy disk drive.

The Personal Computer AT and its compatibles can fit into both groups, however. This computer can have one floppy disk drive, two identical floppy disk drives, or two different density drives. If you have an AT with different floppy disk drives, use the instructions for a single disk drive.

Within each set of directions are separate instructions for microfloppy- and minifloppy-disk users. The separate instructions are given at the end of each section.

During this exercise, you may see a DOS-generated error message on-screen. Owners of systems that have either high-capacity disk drives (the PC XT 286 and Personal Computer AT) or 1.44M floppy disk drives (PS/2 computers) are most likely to see such a message. If you see something quite different from the screen display shown in this exercise, see the section "Understanding DISKCOPY's Error Messages" for additional instructions.

You may want to skim the directions that do not apply to your computer. Then you will be familiar with the proper procedure whenever you use DISKCOPY on a computer with floppy disk drives that are different from your computer's disk drives.

Using DISKCOPY with Two Floppy Disk Drives

The instructions in this section are for users of computers with two identical floppy disk drives. These instructions are for most Personal Computer users and for any PC XT, XT 286, AT, or PS/2 owners whose computers have two identical floppy disk drives.

For the copying process, be sure that the DOS V3 Startup/Operating disk or the DOS V4 Install disk is still in drive A: and that the door is closed. (If you are using minifloppy disks, use the DOS V3 Operating or DOS V4 Install disk instead.) Place a new, blank disk in drive B: and close the door. Then type the following command:

DISKCOPY A: B:

You should see the following message:

```
A>DISKCOPY A: B:
Insert SOURCE disk in drive A:
Insert TARGET disk in drive B:
Press any key when ready. . .
```

The `SOURCE` disk is the disk to be copied, which in this case is the DOS V3 Startup or DOS V4 Install disk. The `TARGET` disk is the blank disk in drive B:. This disk will become an exact copy of your DOS V3 or V4 disk.

If you are using microfloppy disks, or minifloppy disks and DOS V4, you already have the correct diskettes in the correct disk drives, and you can press any key to begin copying. For minifloppy disks and DOS V3, you must first remove the DOS V3 Operating disk from drive A:, place the DOS V3 Startup disk in drive A:, and then press any key to begin copying. If you are using microfloppy disks, you will see this message:

```
Copying 80 tracks
9 Sectors/Track, 2 Side(s)
```

If you are using minifloppy disks, you will see this message:

```
Copying 40 tracks
9 Sectors/Track, 2 Side(s)
```

In 10 to 50 seconds, the light on drive A: will go out, and the light on drive B: will go on. Afterward, you will see the following message:

```
Formatting while copying
```

This message tells you that DOS is preparing the blank disk in drive B: to hold the information that will be transferred from the DOS disk in drive A:.

After DISKCOPY has finished copying the floppy disk, you will see this message:

```
Copy another diskette (Y/N)?
```

If you are using a version of DOS and a disk format that includes additional DOS disks, press Y and repeat the process until all the DOS disks are copied. After you have copied all the DOS disks, press N when asked if you want to copy another.

Skim the instructions in the section "Copying with One Floppy Disk Drive" and perform the steps at your leisure. The instructions for machines with one floppy disk drive work on your computer also, but those procedures are slightly less convenient to use.

Be sure to label and write-protect your new working DOS disks.

Using DISKCOPY with One Floppy Disk Drive

This section is for users of computers that have one floppy disk drive, such as most Personal Computer XTs, XT 286s, Personal Computer ATs, and PS/2s, or for users whose computers have two different types of floppy disk drives. Only one disk drive will be used to copy the disk(s).

At the end of this section, instructions are given first for users whose computers have microfloppy disk drives (who will make one floppy disk copy). Then come instructions for those users whose computers have minifloppy disk drives (who will make two floppy disk copies).

Make sure that your DOS V3 Startup/Operating microfloppy or DOS V4 Install disk is in drive A: and that the door is closed. If you have a single minifloppy drive, insert the DOS V3 Operating or DOS V4 Install disk in the drive. Type the following command:

DISKCOPY A: A:

You should see the following message:

```
Insert SOURCE disk in drive A:
Press any key when ready. . .
```

The `SOURCE` disk is the DOS disk you want to copy, which is already in drive A: if you are using microfloppies; just press the space bar. If you are using minifloppies, remove the Operating disk from drive A:, place the Startup disk in drive A:, and then press the space bar. The light on your floppy disk drive will go on. Users of DOS V4 should leave the Install disk in the drive. If you are using microfloppy disks, you should see this message in a few seconds:

```
Copying 80 tracks
9 Sectors/Track, 2 Side(s)
```

If you are using a minifloppy disk drive, you should see this message:

```
Copying 40 tracks
9 Sectors/Track, 2 Side(s)
```

After 10 to 30 seconds, the following message should appear:

```
Insert TARGET disk in drive A:
Press any key when ready. . .
```

Take your DOS Startup/Operating or Startup disk (or Install disk, if you are using DOS V4) out of the floppy disk drive, put the blank disk into the same drive, close the door, and press a key. You should then see this message:

```
Formatting while copying. . .
```

For DOS V4 users, if the disk is already formatted to 720K, the message will not appear.

DOS is now preparing your blank disk to hold the information from the original DOS Startup disk. After a brief period (anywhere from 20 seconds to about one minute), you may see the following message appear a second time:

```
Insert SOURCE disk in drive A:
Press any key when ready. . .
```

Take out the second disk and again place your DOS Startup disk in the drive. Close the door and press a key. After another brief period, you should see the following message:

```
Insert TARGET diskette in drive A:
Press any key when ready. . .
```

Change disks again, close the door, and press a key. These messages may appear once, or they may appear several times. The number of times the messages appear (and the number of times you exchange disks) depends on the type of floppy disk drive you have and the amount of random-access memory in your computer. Each time the messages appear, simply exchange disks and press a key.

Finally, you should see this message:

```
Copy another diskette (Y/N)?
```

If you are using a version of DOS and a disk format that includes additional DOS disks, press Y and repeat the process until all the DOS disks are copied. After you have copied all the DOS disks, press N when asked if you want to copy another.

Labeling and Write-Protecting Your New Disks

Reminder: *Always label a floppy disk after you perform a DISKCOPY operation; the label reminds you that the disk has something useful on it.*

You now have working copies of your system disks. You can boot DOS from the copy of the DOS V3 Startup disk or DOS V4 Install disk if you want. However, if you have DOS V4, you will see the SELECT program each time. (If you are using 5 1/4-inch disks with DOS V4, you will be asked to put the SELECT disk in drive A:.) You can prevent seeing the SELECT program by typing this command at the DOS prompt:

ERASE A:AUTOEXEC.BAT

This command will remove the AUTOEXEC.BAT file from the disk, and DOS will load just as it does for DOS V3 and earlier versions. You can still use your original Install disk to run the SELECT program later if you want.

Always label a floppy disk after you perform a DISKCOPY operation. The label reminds you that the disk has something useful on it; then you will know not to format it by mistaking the disk for a blank.

If you have microfloppy disks, turn the disk over and locate the sliding tab on the upper left part of the microfloppy disk. Slide the tab up so that you can see through the hole. This action write-protects the disk.

If you have minifloppy disks, you will find a square notch on the right side of the disk. Cover this notch with a special paper or plastic tab called a write-protect tab. These tabs are usually supplied with the disk. You can use any opaque tape or label to cover the notch, as long as you are certain it will not come off easily and possibly "gum up" your disk drive.

Reminder: *Always write-protect disks you have prepared with DISKCOPY.*

Understanding DISKCOPY's Error Messages

Although this section is geared to the exercise example, you can generalize some of these problems and their solutions to any DISKCOPY exercise. If you are using high-capacity disk drives (HC disk drives) and high-capacity disks, this exercise may not work correctly for you. This exercise may also not work on PS/2 computers that use 1.44M disk drives and 1.44M diskettes. If you see any messages indicating that something is wrong (the message may say something about invalid media or about track 0 being bad), press the Ctrl-Break sequence when DOS asks you to place a new disk in drive A:. To do this exercise properly, you need to begin the copying exercise again, using 360K or 720K disks. The format of the source disks will be either 360K or 720K; when these capacities are used in the source drive, do not use HC disks in the target drive.

If you use FORMAT on 360K or 720K capacity disks in 1.2M or 1.44M drives, you will get either a `General failure error` or numerous bad sectors on the destination disk.

Other potential error messages may be one of the following:

```
Unrecoverable read error on drive A:

Unrecoverable write error on drive A:

Unrecoverable write error on drive B:
```

The next line of the error message will state the side and track of the error. If you see the first message, either your DOS disk is bad, or the drive may be faulty. Stop DISKCOPY by pressing the Ctrl-Break sequence. Take the disks to another computer and try the exercise again. If the exercise works, your disk drive probably has problems. If the exercise still does not work, your copy of DOS is probably faulty. Either way, see your dealer about the problem.

If you see the second or third error message (about a write error), the blank disk you are using may be bad, or your disk drive may be faulty. Most likely, the problem is the disk.

Press the Ctrl-Break sequence to stop the program. Repeat the DISKCOPY directions for the kind of disk you are copying. If you are using two disk drives, before you start, remove and reinsert the floppy disk in drive B:. Sometimes a disk does not sit properly on the spindle. Reinserting the disk may correct the problem.

If the same error message appears, repeat the exercise with a different blank disk. If this tactic works, the first disk is bad and should be replaced. If the tactic does not work, you probably have a disk drive problem. Check with your dealer to have the drive repaired.

For a complete look at DISKCOPY's messages, see the Command Reference.

Comparing Diskettes with DISKCOMP

The external command DISKCOMP is a sister command to DISKCOPY. DISKCOMP compares each logical sector of one diskette to the same logical sector of another to determine whether the two disks are identical. DISKCOMP is useful for determining whether a working copy of an original disk has been modified or has errors. DISKCOMP can also determine whether disks containing files transferred by COPY or XCOPY are identical. Due to the nature of first-available, first-used file allocation, however, two disks that contain the same files transferred by COPY or XCOPY will probably not compare without errors.

Cue: *DISKCOMP is useful for determining whether a working copy of an original disk has been modified or has errors; DISKCOMP can also determine whether disks containing files transferred by COPY or XCOPY are identical.*

Like DISKCOPY, DISKCOMP works at the sector level, not at the file level. When a difference is detected by DISKCOMP, only the fact that the compared sectors are different is implied. DISKCOMP does not indicate which disk is "correct" and which is "incorrect". Remember that DISKCOMP is a disk-level command. To compare individual files, use the FC or COMP commands to compare files or sets of files.

Issuing the DISKCOMP Command

The syntax for DISKCOMP is as follows:

DISKCOMP *d1: d2: /1 /8*

d1: is the drive name that holds the first disk of the comparison; *d2:* is the drive name that holds the second disk of the comparison.

The optional */1* switch tells DISKCOMP to compare the disks as single-sided formats only.

The optional */8* switch tells DISKCOMP to compare the disks as 8-sector-per-track formats even though *d1:* (the first disk) has a 9-, 15-, or 18-sector-per-track format. */8* allows comparison of a DOS V1 disk to a V2 or later disk.

Understanding the Operation of DISKCOMP

DISKCOMP determines the format of the first disk and then reads as many sectors from the first disk as available system memory will accommodate. DISKCOMP then examines the second disk's format. If it is compatible with the first disk's format, DISKCOMP compares each sector of the first disk's sectors to each sector of the second disk. This read-and-compare process repeats until the two disks are entirely compared. If sector-comparison errors are detected on the second disk, they are reported. If the second disk has a bad sector marked in its FAT that does not match the first disk's, DISKCOMP will terminate the comparison and report the bad sector.

Reminder: *DISKCOMP reports comparison failures as errors; do not automatically assume, however, that the second disk has file-system errors.*

Because DISKCOMP, like DISKCOPY, works below the layer of DOS that performs file-system bookkeeping, sectors marked as being bad (and therefore not used by DOS for file allocation) may be reported as being part of a bad disk rather than a non-comparing sector. DISKCOMP is blind to DOS's provisions to keep bad sectors from being allocated. Remember that DISKCOMP reports comparison failures as errors. You shouldn't automatically think that the second disk has file-system errors.

Understanding the General Rules of DISKCOMP

Reminder: *Hard disks cannot be compared using DISKCOMP.*

- DISKCOMP will use the default drive for both disks in the comparison if you do not give a drive parameter in the command.
- Hard disks cannot be compared.

- Disk formats of the disks to be compared must be the same (except when using */8*, the 8-sector-per-track override switch).
- Drives participating in a JOIN, ASSIGN, or SUBST command cannot be used in a comparison.
- Network drives cannot be used in a comparison.

Using DISKCOMP To Compare Diskettes

On a system with one 3 1/2-inch drive and one 5 1/4-inch drive, you must use the same drive to compare two like-sized floppies. Assume that drive B: is a 5 1/4-inch drive and that you have two minifloppies to compare. First issue the following command:

DISKCOMP B: B:

DOS will respond with the following prompt:

```
Insert FIRST diskette in drive B:
Press any key when ready . . .
```

Place the first disk in drive B:; then press the space bar once. DOS will report on the first disk's format. In this case, the disk is a 1.2M disk, so the message will read as follows:

```
Comparing 80 tracks
15 sectors per track, 2 side(s)
```

DISKCOMP will read as much of the disk as it can into available RAM and then display the following prompt:

```
Insert SECOND diskette in drive B:
Press any key when ready . . .
```

Remove the first disk from drive B: and insert the second disk in the drive. Press the space bar; the comparison will continue. You will be asked to swap the disks several times. Be sure that you don't mix up the disks.

Caution: *When using DISKCOMP, be sure not to mix up the disks during the swapping process.*

When the comparison process is done and the disks compare, DISKCOMP displays the following message:

```
Compare OK
```

It then prompts with the following question:

```
Compare another diskette (Y/N)?
```

Choose N; DISKCOMP will terminate. Choose Y to compare other disks.

Each track that fails comparison is reported on the screen as follows:

```
Compare error on
side x track yy
```

The actual side and track numbers will appear in place of *x* and *yy*.

Selectively Updating a Disk with REPLACE

The external command REPLACE was introduced in V3.2 as a utility for updating files with different versions of the same files. REPLACE is a real boon for people who must update a disk or directory from another disk or directory.

You can add selectivity to the command with switches and make the updating of the target disk files conditional on date (and time with V4) and the absence of a file. REPLACE even performs an optional search of the target disk directory's subdirectory for a replaceable file.

You can use REPLACE to collect the most recent versions of common files from a group of PCs to a floppy disk. You can use REPLACE to upgrade to a new version of some software package. REPLACE is more versatile than COPY because REPLACE offers conditional copying of source files to the destination. Without switches, REPLACE will transfer files only from the source that already exists on the destination. COPY using the *.* wild card designation transfers all files from the source to the destination.

Issuing the REPLACE Command

The syntax for the REPLACE command is as follows:

REPLACE *d1:path1***filename.***ext** **d2:***path2*\ */switches*

d1: is the optional source drive. If you omit the source drive specifier, the current drive is assumed.

path1\ is the optional path of the source files. If you omit the source path specifier, the current directory is assumed.

filename.*ext* is the required file name and optional extension of the source file(s). Wild cards are allowed in the file name and extension.

d2: is the required target drive specifier where the files will be replaced.

path2\ is the optional target path name where the files will be replaced. If the target path specifier is omitted, the current directory of the specified (or current) target disk is assumed.

/switches are optional switches shown in table 5.4.

Table 5.4
REPLACE Switches

Switch	*Action*
/A	Copies source files that do not exist on the target. Allows *adding* files without overwriting existing files with the same names. /A cannot be used with /S, /D, or /U.
/D	Copies source files with a more recent *date* than files on the target with the same names. /D cannot be used with /A.
/P	*Prompts* for confirmation before copying each file.
/R	Allows overwriting of *read-only* files on the target.
/S	Searches all *subdirectories* of the target directory for a file matching each source file. /S cannot be used with /A.
/U (V4)	Copies source files with a more recent date and time than files on the target with the same names. /U cannot be used with /A.
/W	*Waits* for a key to be pressed before executing REPLACE to allow disk(s) to be changed to the correct source and target.

Understanding the Operation of REPLACE

The external command REPLACE is a useful file-copying command that works at the directory level. REPLACE is included with disk-level commands in this chapter because it lends itself nicely to updating one disk from another.

When issued with no parameters, REPLACE reads the source disk and directory for files matching the command line's source-file specifier. All files meeting the conditions of the source specification are transferred to the destination disk and path. Unlike COPY, REPLACE cannot rename the files as they are copied to the destination; therefore, no file specifier is allowed (or needed) in the destination parameters.

Reminder: *Unlike COPY, REPLACE cannot rename files as they are copied to the destination.*

REPLACE utilizes DOS's standard file allocation method. File space is allocated using the destination disk's FAT, and the destination directory is updated. Files present on the destination disk that are not part of the replacement files remain intact.

If any optional switches are given on the command line, REPLACE performs a compatibility check of the switches to determine disallowed switch combinations. If any are found, REPLACE issues the message `Parameters not compatible` and then terminates (see table 5.4 for switch actions and compatibilities).

If you issue REPLACE with the /W switch specified in the command line, REPLACE will prompt you to press any key before any replacement or adding begins. This *waiting* condition allows you to change disks to those appropriate for the files you want to update.

If the /A, /D, /U,or the /P switches are specified in the command line, REPLACE will test the appropriate conditions of the source file against the destination. If the test passes, REPLACE will proceed with replacing the file. If the test fails, REPLACE will move to the next source file candidate and test again.

If the /P switch is specified in the command line, REPLACE will prompt you before copying the candidate source file to the destination. You can confirm the replacement with Y or reject it with N. The file will not be replaced if you answer N; REPLACE will move on to the next candidate file.

If you use the /R switch to allow replacement of read-only in addition to normal destination files, REPLACE will behave as though the destination file is not read only and replace it. The updated file on the destination disk will not be marked as read-only.

The /S switch allows REPLACE to search all subdirectories of the destination directory for the file to update. You cannot use the /A (add a new file) switch with the /S switch. If /A were allowed with /S, REPLACE would be unable to determine which potential destination subdirectory to add the file to.

The /D switch and the /U switch add time selectivity to REPLACE. PC DOS does not include the /D switch. /D causes REPLACE to update destination files with source files having a more current date. /U causes REPLACE to update destination files with source files that have a more current time and date.

When two or more (allowed) switches are used with the command, REPLACE will consider their affects additive. The actions or selections implied by each switch will be carried out by REPLACE at the point each switch would have been carried out if it were the only switch in the command line.

Understanding the General Rules of REPLACE

- REPLACE is available with DOS V3.2 and later versions.
- REPLACE has no provision to add or update hidden or system files. To transfer system files, use the SYS command.

- You must include a source parameter with REPLACE. The source parameter may be a drive, a path, or a file name and can include all three. The file-name specifier may contain wild cards.
- If the source drive or path name is omitted, REPLACE assumes the default value for the omitted item.
- The destination parameters may include a drive and a path name but may not include a file name. The destination drive or path or both may be omitted. REPLACE assumes default values for omitted destination parameters.
- The /A switch cannot be used with the /U, /D or the /S switches. All other switch combinations are allowed.

Using REPLACE To Update to a New Version of DOS

Cue:
You can benefit from updating to a newer version of DOS.

If you are using an earlier version of DOS, you will benefit from updating to a later version. DOS versions have changed about every two years; it's best not to be more than two versions behind. Of course, if your version of DOS suits you, you may not want to update to a newer version. REPLACE is a command that is well suited to updating the DOS non-system files from new files in a DOS package.

Cue:
Even if you have booted your computer with an earlier version of DOS, DISKCOPY will make exact duplicates of the disk that contains the new version of DOS.

These examples assume that you are updating from a version of DOS that is V2.0 or later. No matter which version of DOS you are using, be sure to first make copies of the original DOS disk(s). You can use DISKCOPY to make these copies.

Replacing your old DOS files with new ones is easy with the REPLACE command. REPLACE can replace the old DOS external command files with newer versions; it can also add commands and other files that do not already exist on a disk. You probably have a directory on your hard disk, for example, that contains all your DOS programs. For the purpose of this discussion, assume that the directory containing the DOS program files is called C:\DOS.

Upgrading to DOS V3.3

To upgrade your hard disk from V3.2 or earlier versions to V3.3, place the start-up disk in drive A: and reboot the computer. Make sure that the date and time are correct; press Enter after confirming each prompt. You will see the A> system prompt. The next step is to replace the current DOS utilities from the old version of DOS with utilities from V3.3. To do so, issue the following command:

```
A>REPLACE A:*.* C:\ /S
```

This command causes DOS to search every subdirectory on drive C:, including the root directory and all subsequent subdirectories, for files that match those on drive A:. If files with matching names are found, the files on drive C: are replaced with the files from drive A:.

Using the /S switch to include subdirectories is recommended so that REPLACE will replace all the appropriate files. You may, for example, have separate directories for DOS utilities, device drivers, and other types of DOS files. REPLACE with the /S switch searches the entire hard disk for matching files.

Heed one caution before you issue this command. If, for some reason, you have other programs or utilities on your hard disk with the same name as the utilities provided with DOS, the existing utilities will be replaced. You may want to verify that you do not have files with the same names as any DOS command. Of course, when you are using REPLACE with the /S switch to replace any files on another disk, you should ensure that a file with the same name as a source file, but with a different content, isn't replaced. Either leave off the /S or rename the potentially conflicting file on the target disk. You can rename the file back to the conflicting name after REPLACE is finished.

The next step is to copy any new files that may have been added with this version of DOS to the hard disk. Use the following command:

```
A>REPLACE A:*.* C:\DOS /A
```

REPLACE examines the names of files in the \DOS directory and on drive A:. If a file exists on drive A: but not in \DOS, the file is copied to \DOS. The /A flag adds files to the destination.

If you are using a minifloppy disk, insert the Operating disk into drive A: and issue the two commands again. This time, however, place the path leading to REPLACE.EXE before the REPLACE command. Use the following commands:

```
A>C:\BIN\REPLACE A:*.* C:\ /S

A>C:\BIN\REPLACE A:*.* C:\DOS /A
```

This technique is ideal for users who keep all DOS files in one or more subdirectories. The first REPLACE command replaces old DOS files with their new counterparts. Because the REPLACE command can examine all subdirectories of the given subdirectory, REPLACE automates the "search-and-replace" mission that is needed. The second command adds the new files to the \DOS subdirectory.

The next step requires the SYS command. Place the Startup disk in drive A: once again and type the following:

```
A>SYS C:
```

This command places the operating system on the hard disk. You do not have to copy COMMAND.COM to the hard disk; it was copied with the first REPLACE command.

At this point, the hard disk is ready with the new version of DOS. Remove the floppy disk from the disk drive and reboot DOS from the hard disk. If all is well, the new DOS boots properly.

Upgrading to DOS V4

Upgrading to V4 of DOS is much easier than upgrading to V3.3. First, insert the V4 Install disk into drive A: and boot your computer. Follow the prompts given to you by the SELECT program and make appropriate responses.

As DOS V4 is being installed, you will need to type the name of the directory you use to contain your DOS files—for example, C:\DOS. This step places the new DOS utilities in the appropriate directory. Old DOS utilities on your hard disk will be replaced with the new utilities. The DOS V4 SELECT program is actually using a version of the REPLACE command to update your hard disk.

The current versions of your AUTOEXEC.BAT and CONFIG.SYS files will be preserved. Two new files will be added—AUTOEXE.400 and CONFIG.400. These files contain suggested statements for your AUTOEXEC.BAT and CONFIG.SYS files. You may want to view the new files and add any new statements to your current AUTOEXEC.BAT or CONFIG.SYS files.

Analyzing a Disk with CHKDSK

The external command CHKDSK analyzes the FAT, the directories, and, if you want, the fragmentation status of the files of a disk. Optionally, CHKDSK will repair problems in the FAT due to lost clusters and write the contents of the lost clusters to files. CHKDSK also provides an option to display all of a disk's files and their paths. Upon completion, CHKDSK displays a screen report of its findings. If you are not familiar with the structure of DOS's FAT and directories or do not understand fragmentation, you should review Chapter 2.

It is good practice to run CHKDSK periodically on your hard disk and important floppies. Because the FAT and the hierarchical directory system work together to manage file allocation, identification and status, a problem in either the FAT or one of the directories is always a serious problem.

Issuing the CHKDSK Command

You issue the CHKDSK command with the following syntax:

CHKDSK *d:path\filename.ext /F/V*

d: is the optional drive name to be checked. If the drive name is omitted, the current drive is assumed.

path\ is the optional path to the directory containing the files to be analyzed for fragmentation. If a path is omitted and a file specifier is given in the command line, the default directory is assumed.

filename.ext is the optional file name and extension for the file(s) to be analyzed for fragmentation. If a file-name specifier is not present on the command line, CHKDSK does not check for fragmentation.

/F is the optional *Fix* switch which instructs CHKDSK to repair any problems encountered.

/V is the *Verbose* switch which instructs CHKDSK to provide file names as the files are being analyzed.

Understanding the Operation of CHKDSK

CHKDSK checks for the following problems in the FAT:

1. Unlinked cluster chains (lost clusters)
2. Multiple linked clusters (cross-linked files)
3. Invalid next cluster-in-chain values (invalid cluster numbers)
4. Defective sectors where the FAT is stored

CHKDSK checks for the following problems in the directory system:

1. Invalid cluster numbers (out of range)
2. Invalid file attributes in entries (attribute values DOS does not recognize)
3. Damage to subdirectory entries (CHKDSK cannot process them)
4. Damage to a directory's integrity (its files cannot be accessed)

CHKDSK then produces a screen report which summarizes disk and system RAM memory usage. Figures 5.2 and 5.3 show typical CHKDSK reports.

Reminder:
Take advantage of CHKDSK's "dry run" capability to assess reported problems on a disk.

Note that you should issue the CHKDSK command without the /F (fix) switch before you subsequently use the /F switch. CHKDSK has the ability to make a "dry run" of its checking routines. CHKDSK with no /F will prompt you if it finds a problem as though you used the /F switch. Take advantage of CHKDSK's "dry run" capability to assess reported problems. After you have assessed the findings of CHKDSK and have taken remedial actions (such as those that follow), you can issue CHKDSK with the /F switch so that the command can fix problems that it finds.

```
D:\>chkdsk

Volume LOGICAL D   created 09-07-1989 11:20p

  33454080 bytes total disk space
     51200 bytes in 18 directories
  13004800 bytes in 766 user files
  20398080 bytes available on disk

      2048 bytes in each allocation unit
     16335 total allocation units on disk
      9960 available allocation units on disk

    655360 total bytes memory
    409856 bytes free

D:\>
```

Fig. 5.2

A typical report produced by CHKDSK with no parameters.

```
D:\>chkdsk \pm\*.*

Volume LOGICAL D   created 09-07-1989 11:20p

  33454080 bytes total disk space
     51200 bytes in 18 directories
  13006848 bytes in 767 user files
  20396032 bytes available on disk

      2048 bytes in each allocation unit
     16335 total allocation units on disk
      9959 available allocation units on disk

    655360 total bytes memory
    409856 bytes free

D:\PM\WIN200.BIN Contains 2 non-contiguous blocks
D:\PM\PM.COM Contains 2 non-contiguous blocks
D:\PM\PM.EXE Contains 2 non-contiguous blocks
D:\PM\PMUSDISK.DCT Contains 2 non-contiguous blocks
D:\PM\CGMMPORT.FLT Contains 2 non-contiguous blocks

D:\>
```

Fig. 5.3

Report produced when CHKDSK is issued with an optional path to the directory to check for fragmentation.

CHKDSK has the ability to read the FAT and directories. DOS's file allocation method gives CHKDSK its method. CHKDSK knows that every file on the disk has its sectors allocated as clusters from the FAT. In addition, CHKDSK knows that each file's directory entry should contain a starting cluster which points to the appropriate entry in the FAT to the file's chain of clusters.

CHKDSK processes each directory starting at the root and following each subdirectory. The indicated cluster chain is checked using the directory entry's FAT pointer. The size of the file in bytes is also compared to the size of the FAT's allocation in clusters.

CHKDSK expects to find enough chained clusters in the FAT to accommodate the file, but not more than necessary. If CHKDSK finds too many clusters, it issues the following message:

```
Allocation error,
size adjusted
```

The file is truncated (excess clusters deallocated) if the /F switch is in effect. You should copy the reported file to another disk before running CHKDSK with the /F switch in case the number of bytes indicated in the directory entry for the reported file is incorrect. If the directory entry has an errant (too low) number of bytes indicated, CHKDSK will effectively "chop off" some of the file. With a copy of the file on another disk, you can recover in the unlikely case the directory entry was wrong.

CHKDSK makes sure that each of the FAT's clusters are allocated only once. In rare circumstances such as power problems or hardware failures, DOS can give two different files the same cluster. By checking each cluster chain for cross-linked files, CHKDSK can report "mixed-up" files. Each time you see the message `filename is cross-linked on cluster X`, copy the file reported in `filename` to another disk. You will find that CHKDSK reports another file with the same message. Copy the second file to another disk also. There's a good chance that the contents of the two files will be mixed up, but you will have a better chance of recovering the files if you save them to another disk before CHKDSK "fixes" the problem.

Unfortunately, if CHKDSK encounters a file allocation chain that loops back to itself, CHKDSK will run in a circle reporting errors. If the /F switch is active, you can easily find your root directory swamped with CHKDSK-produced recovery files. In all likelihood, the action that will terminate CHKDSK when it is chasing a circular allocation chain is the filling of all root directory entries. You can press Ctrl-C to stop the process. If you detect that a circular reference episode is taking place with CHKDSK, don't reissue the command with the /F switch. Try erasing the file and restoring it from your backup.

CHKDSK expects that every cluster in the FAT will either be available for allocation, part of a legitimate directory-based cluster chain, or a marked bad cluter. If CHKDSK encounters any clusters or cluster chains that are not pointed to by a directory entry, CHKDSK issues the following message:

```
x lost clusters in Y chains
```

CHKDSK then prompts as follows:

```
Convert lost chains to files (Y/N)
```

If the /F switch is active, CHKDSK will turn each cluster chain into a file in the root directory. Each created file will have the name `FILEnnnn.CHK` where `nnnn` is a number which increments for each file created by the CHKDSK command's current execution.

You can use the DOS TYPE command to examine the contents of a text file, and you may be able to put the text back into its original file using a word processor. If the file is a binary (program or data) file, the TYPE command won't do you any good. If the problem was with a program file, you may need to use COMP or FC to compare your disk's binary files with their counterparts from your master disks.

The disk does not actually lose any sectors physically. A lost cluster report does not indicate that the clusters are bad. Lost clusters only indicate that DOS made a bookkeeping error in the FAT that makes some cluster or clusters appear to be lost to DOS. The clusters aren't tied to a directory entry, yet the clusters are marked as being in-use. The lost cluster problem is most likely to occur when you are running disk-intensive programs such as dBASE IV or WordPerfect. The programs are not to blame for lost clusters, but the programs increase DOS's exposure to bookkeeping errors. Programs that use disk files to swap sections of program and data too large for memory may ask DOS to read and write work files hundreds of times in an afternoon of computing. Power "glitches" or interruptions, heated hardware components, and electrical interference can turn a cluster chain number into a different number at the critical moment that DOS is writing the number into the FAT. Still, when you consider the millions of bytes that DOS is responsible for in a typical PC, DOS's reliability record is superb. CHKDSK recovers most clusters that bookkeeping errors in DOS lose.

Understanding the General Rules of CHKDSK

- CHKDSK begins execution immediately. If you use a one-disk system, you use the drive B: specifier to allow for changing from the disk containing CHKDSK to the disk to be analyzed.
- CHKDSK will not repair problems found during operation unless you issue the /F switch in the command line.
- You should run CHKDSK at least weekly. Run CHKDSK daily during periods of extreme file activity.
- You must press Enter after answering `(Y/N)?` prompts before CHKDSK will continue.
- CHKDSK may report three hidden files on a DOS system disk. Two hidden files are the DOS system files and the third file is a volume label file.

- CHKDSK will convert lost clusters to files if you answer Y to the `Convert lost chains to files (Y/N)?` by placing the files in the disk's root directory with a `FILEnnnn.CHK` name.

  If CHKDSK terminates because the root directory has no more entries available for converted chain files, clear the current `FILEnnnn.CHK` files by erasing them or copying them to another disk. When the files are cleared from the root, reissue CHKDSK.

Using CHKDSK To Repair an Allocation Size Error

For this example exercise of CHKDSK, suppose that you are copying a group of files from your hard disk to a floppy. During the copy, the lights flicker and then go out completely. In a few seconds, power is restored to normal. Your computer reboots DOS V4 and awaits your input.

Power problems during file operations such as COPY can cause DOS's bookkeeping job to be interrupted. The directory and the FAT can contain errors. To ensure that no errors go undetected, you issue the CHKDSK command on the floppy disk:

CHKDSK A:

CHKDSK begins to analyze the floppy disk in A: and then reports as follows:

```
Volume SCRATCH DISK created 09-12-1989  11:23a
Errors found, F parameter not specified
Corrections will not be written to disk

A:\DBASE1.OVL
Allocation error, size adjusted

    730112 bytes in total disk space
    415744 bytes in 4 user files
    314368 bytes available on disk

      1024 bytes in each allocation unit
       713 total allocation units on disk
       307 available allocation units on disk

    655360 total bytes memory
    409856 bytes free
```

Because you did not specify the /F switch, CHKDSK did not repair the problem. You have the opportunity to examine the problem further before reissuing CHKDSK with the /F switch. You look at a directory listing of the files on A: to

see whether you can determine the nature of the allocation problem. The directory listing shows the following:

```
 Volume in drive A is SCRATCH DISK
 Volume Serial Number is 0000-0000
 Directory of  A:\

DBSETUP  OVL     147968 10-21-88  12:22a
DBASE3   OVL      85024 12-28-88   9:04p
DBASE6   OVL     114832 10-20-88  11:22p
DBASE1   OVL          0 09-12-89   1:46a
        4 File(s)     314368 bytes free
```

The last directory entry, DBASE1.OVL, shows a file size of 0 bytes. A 0-byte file size should never result from a COPY operation. The file's directory size entry is suspicious. To further clarify the nature of the allocation error, you use the CHKDSK report to compare with the directory listing for the same files.

The CHKDSK report shows 415,744 bytes in the 4 files. When you total the bytes in the directory listing, you can only account for 347,824 bytes in the 4 files. Both CHKDSK and the directory listing show 314,368 bytes available on the disk. CHKDSK and DIR both report available disk bytes as the number of bytes in unallocated disk clusters—not the difference between the capacity of the disk and the number of bytes in the disk's files. Both commands get the disk's remaining capacity indirectly from the FAT.

Because both commands agree on the FAT's calculation, you must assume that the directory entry for DBASE1.OVL is incorrect in its reflection of the file's size. CHKDSK can repair the directory entry. Issue the CHKDSK command again using the /F switch:

CHKDSK A: /F

CHKDSK then reports the following:

```
Volume SCRATCH DISK created 09-12-1989  11:23a

 A:\DBASE1.OVL
Allocation error, size adjusted

730112 bytes in total disk space
415744 bytes in 4 user files
314368 bytes available on disk

  1024 bytes in each allocation unit
   713 total allocation units on disk
   307 available allocation units on disk

655360 total bytes memory
409856 bytes free
```

To confirm that the problem you suspect in the directory is repaired, list the directory of the disk again. You see the following:

```
Volume in drive A is SCRATCH DISK
Volume Serial Number is 0000-0000
Directory of  A:\

DBSETUP  OVL     147968 10-21-88  12:22a
DBASE3   OVL      85024 12-28-88   9:04p
DBASE6   OVL     114832 10-20-88  11:22p
DBASE1   OVL      65536 09-12-89   1:46a
    4 File(s)     314368 bytes free
```

Notice that DBASE1.OVL shows 65,536 bytes instead of 0. The available capacity of the disk remains unchanged. Now the total bytes shown in the directory listing is within a few thousand of the difference between the disk's capacity and the bytes free total. You can account for this small difference by considering that some of the files do not completely fill their last allocated cluster. The error in the directory is corrected, and the disk is ready for use again.

Recovering Files and Disks with RECOVER

The external command RECOVER is DOS's "tool of last resort" for file and directory recovery. RECOVER "dismantles" a file or a disk a sector at a time. RECOVER then rebuilds as much of the file or the disk as it can recover. The need for a RECOVER command stems from the possibility that a disk will develop bad sectors in a file or directory.

CHKDSK works with the FAT and root directory; RECOVER works with one file or with a complete disk of files and subdirectories. When CHKDSK can't complete its processing because of a damaged directory, you can use RECOVER. When a program or a data file has DOS-reported problems, you can use RECOVER to salvage as much of the file as RECOVER is able. As always, the best defense against file and directory damage is making full and incremental backups. The BACKUP and RESTORE commands are discussed in Chapter 7.

Caution:
Think of RECOVER as a "last-resort" command whose job is to reduce the total damage to directories and files.

The syntax form of RECOVER that recovers a complete disk will dismantle all of the files and directories on the disk. You should never experiment with the form of RECOVER that works on complete disks unless the disk is unimportant.

Once RECOVER has "fixed" the disk or directory, you may spend hours trying to recover from RECOVER. Don't get the idea that RECOVER is a weak command or that it is an incomplete command. It would be more proper to think of RECOVER as a "last resort" command whose job it is to reduce the total dam-

age to directories and files. The form of RECOVER you use on a single file is not risky, however. Each form of RECOVER is covered separately.

Issuing the RECOVER Command

RECOVER has two general forms:

1. To recover a single file use this syntax:

   RECOVER *d:path/***filename**.*ext*

2. To recover a disk, use this syntax:

   RECOVER *d:*

Each form is covered in more detail in the following sections.

d: is the drive holding the disk to be recovered. In the disk recovery syntax, the drive name is required. In the file recovery syntax, the drive is optional.

path/ is the optional path to the file to be recovered in the single file recovery syntax. If the path is omitted in the file recovery syntax, the current directory is assumed.

filename.*ext* is the mandatory literal file name and the optional extension in the single file recovery syntax. If a file name is present, the single file mode of RECOVER is assumed. Wild cards are not permitted.

Using RECOVER on a Defective File

As its name implies, RECOVER enables you to recover a file that contains defective sectors. Defective sectors may develop on a disk due to wear and tear, loss of magnetic fields, electronic problems in a drive, or other problems. In any event, one sector can make an entire file inaccessible to DOS.

The recovery, however, is partial. With RECOVER, you can make a new copy of a file, minus the data held in the bad sectors. Depending on the file being recovered, you may or may not be able to use the recovered file. If the file is a program file, you are almost guaranteed that the recovered program will malfunction. If the file is a data file, especially a text file, you may be able to use its contents. Even if the file is not usable after recovery, rest assured that the bad sectors are marked as being defective in the FAT so that they will not be used again.

Issuing RECOVER To Recover a Single File

The syntax for the file form of RECOVER is as follows:

*dc:pathc***RECOVER***d:path***filename**.*ext*

dc:pathc\ is the optional disk drive and path name to the RECOVER command.

d: is the optional disk drive name if the file is not on the current disk drive.

path\ is the optional path name if the file is not in the current directory.

filename is the root name of the file to recover, and *.ext* is the optional extension.

After you use RECOVER on a file, the file has the same name as before. The bad sectors on the disk are "removed" from use, but the data in them is lost. This recovery method reduces the total capacity of the disk by the number of bytes contained in bad sectors that have been removed.

Caution: *For practical reasons, use RECOVER only on text and data files.*

For practical reasons, the only files you should recover and try to use are text and data files. Do not bother recovering program files. When information is lost from such a file, the program may not run or, worse, the program may run erratically and do some file damage on its own. Copy a backup copy of the program to the disk after you recover and erase the damaged file. You can then use the new program. Do not forget to erase the damaged file if you don't have access to a backup copy of the program.

You will need to edit text or data files after you recover them. For text or ASCII data files, use a text editor to remove any spurious characters or to add lost information. You may need special programs to restore non-ASCII (binary) data files to their original states. Many applications programs will refuse to read data files that have been recovered because essential header or internal pointer information may have been deleted.

There is little reason to be in the position to rely on RECOVER. If you make frequent backup copies of your hard disk or disk copies of your floppy disks, you can usually restore files easily from the backup copies and then reedit as necessary. You should run RECOVER first, though, to "hide" the bad sectors on the disk.

The only way to remove bad sectors is to reformat the disk. If the disk is physically damaged, formatting will not help, however. The magnetic coating from the damaged area will cause the drive's heads to do further physical damage. If the disk is a floppy disk, you should retire it. Hard disks, however, are difficult to retire. You will have to live with any flaws unless the hard disk is physically damaged.

Bad sectors are a bad sign. Either the magnetic coating on the disk is damaged, or you have a problem with the electronics of the disk drive. The most likely reasons—in order—are these: (1) mishandling of the floppy disk or hard disk; (2) physical wear from normal use of the floppy disk (seldom applicable to a hard disk because its recording heads do not touch the surface when the drive is in use); (3) mechanical or electronic failure of the disk drive; (4) electronic failure of the disk interface or controller card (the board inside your computer); (5) a damaged disk drive cable; (6) bad RAM (random-access memory) in the computer.

Bad RAM occurs infrequently. But when it does, the change can cause erratic or disastrous disk performance. The best you can hope for is that DOS will just go "dead in the water." The worst that can happen is that DOS will put garbage in the FAT, directories, and your files and you will have to use CHKDSK and RECOVER. That is why a memory-parity error makes your computer "lock up." PC designers wisely chose to stop the computer so that only work in progress is destroyed and not your previous work, programs, and data stored on the disks.

If a floppy disk is physically worn, the time has come to retire it. Look through the access holes at the floppy disk's surface. Dark grooves indicate wear. Dark or dull splotches may indicate spots where the floppy disk has been mishandled or contaminated. If you see a fold or crease, retire the disk.

Mechanical or electronic problems are more difficult to analyze. Run your diagnostics program. For floppy disk drives, use a disk formatted on a drive you know is good. If the problem is in the hard disk, back the disk up immediately!

If problems show up when you run the diagnostics, have your computer or disk drive repaired right away, before other floppy disks are harmed. If a problem with a disk drive shows up in the diagnostics, it may be caused by the disk drive, the interface board inside the computer, the cable that connects the board and the disk drive, the power supply of the disk drives, or possibly something else in the computer's main circuitry. Whatever the problem, have it repaired immediately.

Seldom do you electronically damage a floppy disk or hard disk when you read information. If the problem is not in the floppy disk itself, and you have not written information to it, the floppy disk and its files are probably still intact. There is always a small chance that the disk drive wrote some garbage instead of reading information. Check out all the files or floppies you have used since you first detected the problem. Make sure that the information is intact.

Using RECOVER on a Damaged Directory

Bad sectors can develop in directories. This problem is a serious one. If a bad sector is in the root directory, the problem is grave. The root directory is in a fixed position on the hard disk or floppy disk. Because DOS cannot move the root directory, this condition can be "fatal." The second form of RECOVER, the potentially dangerous form, recovers damaged root directories.

If a subdirectory has bad sectors, the problem is serious, but not fatal. CHKDSK can cope with it. Before you run CHKDSK, take the following steps for copying the floppy disk or hard disk. If CHKDSK does not work, RECOVER is your final choice.

Taking Preliminary Steps

First, you should copy or back up all the files from the problem disk that you can. Use a different set of floppy disks than the set you used for your last backup copies. Your last backup set is important insurance. Keep your last backup floppy disks intact. You may need to use them if all else fails.

For a faulty floppy disk, use the COPY command to copy each file to a second floppy disk. Then use the DISKCOPY command to copy the bad floppy disk to a third floppy disk. If the offending disk is your hard disk, use a different set of backup floppy disks. Run BACKUP, keeping your master backups and last daily backups intact.

Because you do not know at this point the cause of any bad sectors, you should suspect that the same problem exists in the backed-up or copied files you have just made. Whatever caused the directory to develop bad sectors may have damaged other areas on the floppy disk or hard disk. Make sure that the files you just copied or backed up are correct before you fully trust them.

(If the problem is in a subdirectory, use CHKDSK first. If CHKDSK does not work, use the disk-level form of RECOVER.)

Issuing RECOVER To Recover a Directory

The next step is to use RECOVER. The syntax for this form of RECOVER is as follows:

*dc:pathc***RECOVER** *d:*

dc:pathc\ is the optional disk drive and path name to the RECOVER command.

d: is the optional disk drive name. If the current drive does not hold the disk you want to recover, give the appropriate disk drive name.

DOS will run through the file allocation table (FAT). Remember that the FAT knows the disk clusters (sectors) where each file is stored. But the FAT does

not know the previous file name, the file's attributes (system, hidden, and so on), or the file's date and time because these details were lost in the defective directory.

DOS first re-creates the root directory. Then DOS begins to create files with the name `FILEnnnn.REC` in the root directory. `nnnn` is a number from 0000 to 9999. Each file that DOS creates represents one of the recovered files from the disk. Every file on the disk becomes a `FILEnnnn.REC` file: program files, data files, and subdirectories.

Identifying the Recovered Files

Now your detective work begins. Each `FILEnnnn.REC` can be anything—a normal file or a subdirectory. You must find which `FILEnnnn.REC` files hold the information you need to keep and which `FILEnnnn.REC` files (such as subdirectories) hold information you can discard. The files' previous names are lost. Their dates and times are also lost. The best indicator of what is in each file is the content of the file itself. A regular printout of the output of TREE would remind you of the file names.

You will need several tools to help you. The TYPE command can display on-screen the characters in a file. This command will help you identify ASCII text files. Program files and subdirectories are different, however. Most of their information is displayed as gibberish. To display the contents of these files, you will need DOS's DEBUG, NU from the Norton Utilities, DUMP.EXE from Phoenix Associates, or a similar program.

With one of these tools, you can find the files you were not able to copy or backup. Copy these files to another floppy disk and change their names back to what they were. Make sure that the files are intact. There is a small chance that whatever caused the directory to develop bad sectors may have affected other areas of the floppy disk or hard disk, making other files bad also.

Before you use RECOVER, you might want to practice first on a copy of any floppy disk. Use DISKCOPY to replicate the problem on another disk. You'll be able to see how RECOVER works without risking the disk you want to recover. Then work with a copy of the disk (not the original) you are trying to recover. If you botch up the copy, you still have the unaltered copy and nothing is really lost except time.

Cue:
Before using RECOVER on a directory, practice first on a copy of any floppy disk.

When you try to recover a floppy disk with a flawed directory, again use a copy. If you make a mistake recovering the copy, you can make another copy of the original, flawed floppy disk. If you work with the flawed floppy disk and do not have a copy, a mistake can be costly.

Remember that this discussion does not apply to the RECOVER command which is used on a single file. Because this individual file process works on one file at a time, the worst that can happen is that you will lose part of the file.

After you have re-created your files, you should reformat your floppy or hard disk. FORMAT gives a message when it cannot properly format any system area, including the areas for the boot record, the root directory, and the FAT. If you use FORMAT's /S switch, the system files' areas are also checked, and DOS gives error messages if these areas are bad. The error messages indicate that the floppy or hard disk is currently unusable. You can retire a floppy disk, but you must have a hard disk repaired or purchase a new one.

Remember to back up your data frequently. You will have less work and experience less frustration if you back up your floppy disks and hard disk instead of trying to re-create them with RECOVER.

Understanding the General Rules of RECOVER

- The single file form of RECOVER retains the recovered file's name and directory entry.
- The disk form of RECOVER files and directories are recovered to the disk's root directory with `FILEnnnn.REC` file names.
- Before RECOVER begins execution, you see the prompt `Press any key to begin recovery of the file(s) on drive X`. `X` is the drive that will hold the disk to be recovered.
- RECOVER will terminate execution when the root directory of the recovered disk fills with `FILEnnnn.REC` entries. If the root directory fills, RECOVER will warn you with the following message:

```
Warning--directory full
```

- If you issue RECOVER with a path specifier and no file name specifier, RECOVER will respond `File not found`.

Summary

The disk-level DOS commands covered in this chapter provide you with a means to manage and maintain the disks and drives that you use with your PC. The following key points were covered:

- ❑ FDISK manages DOS partitions in the hard disk partition table.
- ❑ FORMAT initializes disks and places DOS bookkeeping tables on disks during formatting.
- ❑ LABEL adds and changes a disk's volume label text.
- ❑ VOL displays a disk's volume label text.

- SYS transfers DOS system files from one disk to another.
- DISKCOPY replicates one floppy disk to another floppy disk with the same format.
- DISKCOMP compares two floppy disks on a track-by-track basis and reports comparison errors.
- REPLACE selectively updates a disk or directory's files from another disk or directory's files.
- CHKDSK analyzes a disk's FAT, root directory, and optionally, file fragmentation. With the /F switch, CHKDSK repairs defective directory entries and the FAT.
- RECOVER salvages directories and files that contain bad sectors on a disk. RECOVER has one form of syntax for a file and another form of syntax for a disk.

In the next chapter, you will learn about commands that manage disks at the directory and subdirectory level. These important commands are your tools to structure your individual version of the DOS hierarchical directory system.

6

Managing and Navigating the Directory Hierarchy

One of the primary features of DOS is its hierarchical directory system. Chapter 2 explains how DOS uses directories in conjunction with the file allocation table (FAT) to log the physical location of files on a disk. DOS uses internally related directories and the FAT to provide a framework for the management of the hierarchical file system.

Working with DOS directories, however, does not necessarily give you a clear view of this internal file management. Ideally, as long as DOS provides you with the commands to organize directories, you shouldn't be bothered with what is going on behind the scenes. You simply learn the details of the commands. Understanding some of these internal details, however, will help you get the most from any discussion of the DOS directory system. This chapter presents some of these internal details of file system management.

The chapter's main focus, however, is the external view of the file system. This is the view you have of directories when you work with applications programs and DOS commands. From this external point of view, a freshly formatted disk is ready for you to give it a tree-structured file system any way you see fit. DOS directory commands let you design, manage, and modify your directory tree.

Many DOS commands deal with directories in some way or another. For example, you can copy a file from one directory to a different directory with COPY. Yet COPY is a file-level command because it deals primarily with files. The directory commands, on the other hand, have a definite connection with directories. This chapter explains how you can design, manage and modify your directory system based on the directory-level commands provided with DOS. The commands presented in this chapter include the following:

- MKDIR (MD)
- CHDIR (CD)
- PATH

- APPEND
- TREE
- XCOPY
- RMDIR (RD)

The chapter also offers some suggestions for efficient management of the hierarchical directory system on your hard disk.

Understanding the Hierarchical Directory System

A hierarchy is an organization of entities. Entities can refer to people, objects, ideas, files, or many other things. Coins, for example, can be arranged by denomination, and soldiers can be organized by rank. There are, of course, many ways to organize entities besides hierarchically. Stones in a driveway are arranged randomly; bricks in a wall are arranged in parallel rows; and telephone book listings are arranged in alphabetical order. The arrangement of the stones, bricks, and listings is not hierarchical. What makes hierarchical organization useful is that its entities are arranged by some method of relationship into branching series of dependencies.

To a genealogist, entities may be people in a family tree. To DOS, entities are directories in a directory system. In either case, the hierarchy begins with the essential core or root entity. In a family tree, the core entity might be great-great-grandfather John Materson. In DOS, this core entity is the root directory. In genealogy, people can trace their roots through their parents and their parents' generations. In DOS, subdirectories can trace their paths back to the root directory. It's easy to see why the DOS hierarchical file system is called a tree-structured file system. At the base of this tree structure is the root directory.

Understanding the Directory Entry

Cue:
The DOS FORMAT command automatically creates a root directory on any disk.

Every DOS-created disk has a root directory, which is automatically created by the DOS FORMAT command. The root directory is a table, fixed in length, which is placed in the control area of each disk. As part of the file-allocation process, DOS makes directory entries consisting of file names and locations as well as status information about each file. Figure 6.1 shows the main components of each directory entry. The directory entry is DOS's essential file identifier.

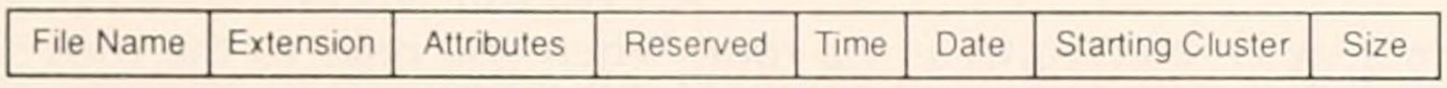


Fig. 6.1

A directory entry.

The directory entry is not the file itself. Rather, the entry is a description of the file in DOS's own annotation. DOS uses this description to locate a file's content, manipulate the content, and provide you with information about the file.

Fields in a directory entry store key information about a file. Figure 6.1 displays the field of a file entry. The 8 characters of the file's root name are stored in the root-name field; the 3 characters of extension are stored in the extension field. Subdirectory names are also stored in the root name and extension fields. Volume labels use the root-name and extension fields as an 11-character volume-label field.

The attribute status for a file is stored in the attribute field (see fig. 6.1). The attribute field is one byte long; each attribute uses one bit for the attribute indicator. If the bit is set to on, the attribute is true. Two of the eight attribute bits are reserved for future use. The remaining six attributes are used by DOS. The file attributes include the following: archive, subdirectory, volume label, system file, hidden file, and read-only file.

Following the attribute field of the directory entry is a reserved field. This field is not used by DOS but reserved for future use. Next are the time and date fields. The time field stores the time that the file represented by this entry was created or last modified. DOS stores the time as hours, minutes, and seconds. The date field stores an encoded date indicating when the file was created or last modified. DOS stores the date as the year (relative to 1980), the month (1-12), and the day (1-31).

Following the date field, DOS stores the starting cluster number for the entry's chain of allocated clusters. The starting cluster number points to the entry in the FAT that begins the file's cluster chain. Without this starting cluster field in the directory entry, DOS has no link to the file's physical storage locations on the disk.

The final directory entry field is the file size field, which stores the actual number of bytes contained in the file. The file size is normally different from the size of the disk space allocated to the file because DOS allocates space in cluster units. Cluster units are multiples of a 512-byte sector. A 1-byte file will show 1 byte of file size in the directory entry, but the file will use at least 512 bytes of disk storage space. You can see that the directory entry DOS makes for a file provides a great deal of information for DOS's bookkeeping functions.

Understanding the Root Directory

Cue:
The root directory is designated in the command line by a backslash (\).

The root directory is referred to as *the root*. When you refer to the root directory in a command line, however, you designate it by using the backslash symbol (\). The root directory has no given name. It is always called the root.

Every DOS disk has a root directory, so to differentiate between the roots of different disks, DOS recognizes drive names. The designations **A:**, **B:**, and **C:** each name the root directory of a different disk. As you recall from Chapter 2 and the discussion of FDISK in Chapter 5, hard disks can be partitioned into more than one logical drive, and each logical drive has its own root directory.

Cue:
DOS uses certain files and directory entries for its own internal purposes.

A file is a part or by-product of your computing work. Normally, DOS enables you to control a file. You have access to a file; you can manipulate it. If the file is a program file, you can execute it. On the other hand, DOS uses some files and directory entries, such as system files, volume labels, and subdirectories, for its own internal manipulation. Special files are on the disk because DOS puts them there, not necessarily because you want them there. In a discussion of hierarchical directories, the subdirectory file is an important special file.

When you add the first subdirectory to the root directory, you are introducing a hierarchical directory structure on the disk. The root directory now has an offspring, and DOS will manage your creation. Without your overview, DOS prepares a special subdirectory file and gives it the same basic capability to receive file entries as its parent directory, the root.

Understanding Subdirectories

A subdirectory is a file that DOS uses in the same general way that it uses the root directory. Recall from Chapter 2 that the root directory is a fixed-length directory table in the control area of each disk. Subdirectories share the common 32-byte entry size used to record information about a file with the root directory. You can think of the root directory and subdirectories as being just "plain" directories because most of your DOS commands use subdirectories and the root directory in a similar way. For a discussion of DOS commands, it is nearly always as meaningful to say "directory" when referring to the root or a subdirectory as it is to use "root" or "subdirectory."

When working with files, you can easily get the impression that DOS keeps the files themselves rather than entries about the files in directories. This is an erroneous impression. There is no physical division of a disk into subdirectories of files. A directory's files may be spread randomly across the disk.

Yet it is logical to think of a file's data as being in a particular directory because a file's data is tied to an entry in that directory. You give directory parameters in commands by including paths. When most commands with path parameters

carry out their actions, the objects of the actions are files. DOS does a good job of hiding from users the random locations on disk of most files by keeping a hierarchical order in the directories.

For the purpose of discussing the contents of a directory, when the contents are files—not file entries, thinking of a directory as a "place" on the disk does no harm.

Specifying a Path in a Command Line

When DOS was introduced, the commands that dealt with files allowed drive specifiers and file specifiers—but not directory specifiers. DOS V1 did not need to know how to find the directory that contained the entry for a desired file. V1 had just one directory per disk. That directory was to become V2's root directory. Even with this subdirectory capability, DOS's default logging kept at the user's discretion the issuance of commands with directory specifiers. Many floppy-only users issued the same commands with the same parameters as they did when they were using V1. By the time DOS V3.0 was introduced, hard disks were in common use, and the subdirectory capabilities that hard disks justified finally caught on with DOS users.

Dealing with subdirectories therefore introduced a new element in syntax lines—the directory specifier or parameter. Because DOS logs to the root directory as the default directory, file commands without directory parameters look like DOS V1 commands. If you do not use subdirectories, the directory specifier is not necessary.

In a symbolic form, a command line that includes the directory "directions" as an element of syntax looks like the following:

COMMAND *drive:path directory_name\full_filename*

In this syntax line, **COMMAND** is the symbol for the DOS command.

drive: is the drive specifier—the symbol for the name of the disk drive holding the disk this command will use.

path is the path specifier—the symbolic place holder for the chain of related directories that lead to the final element of the syntax—the directory name or the file. *path* is optional in the command line. If you don't supply a path, DOS uses the path to and including the current directory of the indicated (or default) drive as a default value for path. DOS V1 did not have this element of syntax. V2 and later versions include the path element of syntax in many commands.

Cue: *The path specifier in a syntax line symbolizes the chain of related directories that lead to the object of the command's action.*

directory_name—the directory specifier—and/or *full_file_name*—the file specifier—are the objects of **COMMAND**'s action. Commands that manage files (such as COPY) may have sets of full file names (file names plus extensions) to

reflect source and target parameters. Commands that manage directories (such as MKDIR) have a *directory_name* as the object of the command. This chapter will discuss many commands that use directory names as the objects of the actions of commands.

The notion of using a path in a command line can be a slippery concept to grasp. You can think of the path as the listing of every directory required to get to a file. In this context, examine the following command:

DIR C:\DIR1\DIR2\FILE.111

The command is DIR, the drive specifier is C:, and the path specifier is \DIR1\DIR2. The final \ delimits the path from the file name on which DIR is reporting—FILE.111.

The symbolic syntax of this command is as follows:

DIR *d:path\filename.ext*

In the DIR example, the path is a listing of directories that DOS needs in order to find FILE.111 and make its report. The path points to the object of the command.

When the object of the command is a directory name rather than a file, the syntax representation is different. As an example, look at the the subdirectory management command MKDIR. To make a directory called DIR2 as a subdirectory of DIR1, you issue the following command:

MKDIR C:\DIR1\DIR2

The command is MKDIR, the drive name is C:, and the path is \DIR1. If the directory name DIR2 is the object of the command's action, the directory name is listed in a syntax line as a separate element called *directory_name*. If DIR2 is a syntax element called *directory_name*, \DIR1 is the path that leads to the point where the MKDIR command is to carry out its action—adding the DIR2 directory. MKDIR's syntax representation would then look like the following:

MKDIR *d:path\directory_name*

Reminder: *Read the text that accompanies the syntax lines for an explanation of each syntax element.*

Good syntax representation is always accompanied by an explanation of each element of the syntax. If you have any doubts about how to form a command line with a path included, read the explanation of each element of the syntax. To make path specification more meaningful, you can follow three guidelines for specifying a path:

- If the path specifier begins with the backslash character (\), DOS interprets the path as being from the root.
- If the path specifier does not begin with \, DOS interprets the path as being relative from the current directory.

- If the path specifier is omitted, DOS assumes that the path is the path of the current directory and that you are supplying the object of the command's action in a command-line parameter. The object may be a file name and extension, or in the case of the directory commands, a directory name.

Using PROMPT To Display a Full Path

Cue:
The DOS PROMPT command displays the full path.

While you are reading about directory-level commands, you might want to confirm that your DOS prompt is displaying the current full path (drive and path) of your current directory. To give you visual confirmation of your current logged drive and any logged directory, DOS provides the PROMPT command to display the full path as part of the DOS prompt. Chapter 8 details the PROMPT command fully, but for now, enter the following command at the DOS prompt:

```
SET
```

DOS replies with the current setting of the prompt string. Look at the string and determine whether the string contains the characters pg.

If the string contains these characters, you can see the current drive and path in the DOS prompt. If the string does not contain these characters, you may want to change the prompt string so that you can see the current full path. To do so, issue the following command:

```
PROMPT $p$g
```

As soon as you reboot, the prompt returns to its former look. If you want to include the path as a permanent part of your DOS prompt, read the section on PROMPT in chapter 8.

With the prompt established as pg, you can log, for example, to the \DOS directory of drive C: and see the following prompt:

```
C:\DOS>
```

Making Directories with MKDIR (MD)

Because the only automatically available directory on a DOS disk is the root, any additional directories must be added by the user. DOS provides the MKDIR, or MD, command which you use to tell DOS to make a new directory. The MKDIR and MD names for this command are identical in operation. You can use either name for the command.

Cue:
The MKDIR (MD) command creates a directory.

When you issue the MKDIR command, you are instructing DOS to add a new subdirectory to some part of the file system tree.

Issuing the MKDIR Command

Remember that any time you can use MKDIR, you can use MD in its place. The following syntax and examples use the MKDIR version of the command. The syntax for the MKDIR command is as follows:

MKDIR *d:path***dir_name**

d: is the optional target drive specifier containing the disk where the new directory is to be added. If the d: parameter is omitted, the default drive is used.

path\ is the optional path specifier to which the new directory will be added.

dir_name is the name for the new directory; a directory name must be given.

Reminder: *The length of the full path (drive, path, and new directory name) must not exceed 63 characters, including delimiters.*

The length of the full path (drive, path, and new directory name) must not exceed 63 characters, including delimiters. The full path name is the measure for length. If you are creating a directory relative to your current directory, you must count your current directory's full path length from the implied root \ character as being part of the 63-character limit.

For special considerations concerning the path specifier in MKDIR and other commands, see the section "Specifying a Path in a Command Line" in this chapter.

Understanding the Operation of MKDIR

When you issue the MKDIR command, DOS does some work behind the scenes. First, DOS verifies that the given (or implied by default) parent directory exists. Then, DOS confirms that the new directory has a unique file name. To confirm uniqueness, DOS looks in the proposed new directory's parent directory to ensure that the name is unique. The parent directory cannot contain an entry for a file that has the same name as the proposed directory's name. Because subdirectories are files, DOS has no convenient means of separating a directory from a normal file with the same name. When creating a directory, a file name can be a possible naming conflict.

After confirming that the new directory name is unique in the parent directory, DOS allocates a file for the new directory and marks (as a subdirectory) the new file's entry in the parent directory. When the new directory file is allocated, the directory is technically created. MKDIR is not yet finished making the new directory, however. The new directory must be linked to its parent in another way.

DOS completes two entries in the new directory. The first entry is that of the new directory's information, which includes the new directory's starting cluster number in the FAT. The first directory entry's name is always . (pronounced

"dot") regardless of the true name of the new directory. Except for the name difference, the dot entry in a subdirectory is the same as the subdirectory's entry in its parent directory.

The second entry is that of the parent directory's information, which includes its starting cluster in the FAT. Every subdirectory, except those subdirectories whose parent directory is the root directory, has a valid starting cluster which points to the subdirectory's parent directory. The name given to this entry in the new directory is always .. (pronounced dot-dot). DOS can use this dot-dot entry to find the current directory's parent directory regardless of the name of the parent directory.

Because these two constant named entries point to variable, named directory files through the FAT, they are called the *alias* files. The alias entries are directory entries that don't "own" the files they point to. Disk-level "analyze-and-fix" commands such as CHKDSK and RECOVER know that these alias entries point to starting clusters that other directory entries point to at the same time. CHKDSK and RECOVER are not tripped-up by this seeming cross-linking of clusters.

Reminder: *The . and .. directory entries are also called the alias files.*

The usefulness of the aliases may not be immediately obvious. Consider that COMMAND.COM accepts the aliases as legitimate path specifiers. By using the dot-dot alias as a path specifier, you can access the parent directory of your working directory no matter at what point in the directory tree your working directory is located. (This same shorthand mechanism is available to DOS for internal use as well.) You will see examples of aliases in commands in this chapter.

Reminder: *DOS accepts the . and .. directory entries as legitimate path specifiers.*

Understanding the General Rules of MKDIR

- MKDIR and MD are different names for the same command. Both commands produce identical results.
- All characters allowed in a normal file name can be used in a directory name.
- A directory name can include up to eight characters and can be optionally followed by a period (.) and up to three characters of extension. Most DOS directories, however, do not include extensions.
- A directory name cannot be the name of a standard DOS device such as AUX, PRN, and CON.
- A directory name cannot duplicate any file name in the intended parent directory. The duplicate can be a normal file, a hidden file, a system file, or a volume label.

- The specified or implied parent directory of the intended directory must exist. MKDIR cannot make more than one directory at a time. On a disk with only a root directory, the command **MKDIR \PROG\DATA** will not work.

Creating a Directory with MKDIR

For this example, assume that you are currently logged on the root directory of a newly formatted floppy disk in drive A:. To create the directory \DATA on the disk in drive A:, enter the following command:

MKDIR \DATA

DOS will make the new directory—DATA—on the disk in drive A:. To confirm the new directory's existence on A:, you can enter the CHKDSK command from the DOS directory of your hard disk for a report of the disk in drive A:

C:\DOS\CHKDSK

DOS displays the following:

```
730112 bytes total disk space
  1024 bytes in 1 directories
729088 bytes available on disk

655360 bytes total memory
603152 bytes free
```

Notice that the available space of this disk has been reduced by 1K because DOS allocated a two-sector cluster to the newly created \DATA directory.

You can achieve the same result by issuing the following MKDIR command while logged to the root directory:

MKDIR DATA

Cue:
If you do not specify a path with the MKDIR command, DOS makes the current directory the parent directory of your new directory.

The directory is created directly from the root even though no disk name or beginning backslash is given in the command line. DOS uses the current (default) drive, drive A:, and makes the directory relative to the current directory, the root. If you do not specify a path, DOS makes the current directory the parent directory of your new directory.

Changing the Current Directory with CHDIR (CD)

Cue:
Use the CHDIR (CD) command to change the current directory.

DOS remembers the current directory path as well as the current drive name. You issue the CHDIR or CD command to change the current directory to a new current directory. The directory you make current with CHDIR is the directory that DOS will search first to find files. Both CHDIR and CD initiate the same command; their usage is identical. The CHDIR command is the hierarchical directory system's navigation command. It is the command you use to position yourself in different working directories.

If you use the CHDIR command with no path specifier, DOS will report the current directory's path name for the specified or default drive on the display. CHDIR alone (with no parameters) is the "Where am I?" command. Issuing only CHDIR tells DOS to report where commands using no drive and path specifiers will do their work. Using only CHDIR in the command line does not change your current directory.

Issuing the CHDIR Command

The syntax for CD or CHDIR is as follows:

CHDIR *d:path*

d: is the optional drive name specifier. Use of the CHDIR command with a drive specifier and no path specifier will not log the default drive to the drive in the drive specifier.

path is the name of the directory you want to make the default directory on the disk in drive *d:*. If no drive is specified in the command line, the path specifier is assumed to be on the current drive. The path specifier can begin with the .. alias.

Understanding the Operation of CHDIR

DOS stores the name of the current directory in an internal location. When you boot your PC, DOS makes the root directory the default, and CHDIR stores the new directory name internally. If you omit the path, and DOS must provide a path for a command line, DOS receives the current value of the storage area.

Reminder: *Each drive that DOS recognizes on your system has its own current directory value stored by DOS.*

Each drive that DOS recognizes on your system has its own current directory value stored by DOS. When you issue CHDIR with a path specifier, but no drive specifier, DOS first consults the logged drive storage area to determine which drive is the current drive. When DOS knows the current drive, DOS verifies that the path from the CHDIR command exists. If the path is not found, DOS issues the `invalid directory` message and does not change the current directory, which is stored internally. If the path does exist, DOS stores the path's name in the internal storage area as the current directory.

If the CHDIR command is issued with no parameters, DOS determines the logged disk internally, finds the name of the logged disk's current directory, and then displays the current directory. CHDIR issued with only a drive parameter will report the specified drive's current directory.

Understanding the General Rules of CHDIR

- CHDIR and CD are different names for the same command. Both names produce identical results.
- The drive and path specifiers in the command line must be valid.
- If the path specifier begins with a \, CHDIR assumes that the directory specified in path is absolute from the root.
- If the path specifier does not begin with a \, CHDIR assumes that the directory given in path is relative from the current directory.
- When the path specifier is omitted from the command line, CHDIR reports the current working directory for the disk in the drive specifier.
- When the path specifier and the drive specifier are omitted from the command line, CHDIR reports the current working directory of the logged drive.
- When only a drive is specified in the command line, CHDIR will report the current directory of that disk, but will not log that disk as the current disk. CHDIR with no path parameter is always a reporting command and does not change any logged values.

Using the Two Forms of CHDIR To Change Directories

CHDIR has two operational forms. One form returns a report of a disk's logged directory's path name. The second form of CHDIR replaces a disk's logged directory with the directory given in the command line. The action of the command in the two cases is identical.

Suppose that the current directory on drive A: is \DIR1. To receive a report of the logged (default) directory of a disk, you issue the CHDIR command in the following form:

CHDIR A:

CHDIR reports the following:

```
A:\DIR1
```

Suppose that the logged disk is C:, and that the current directory of C: is \KEEP. To see the default directory or working directory (the one where DOS will carry out commands if no drive and path specifiers are used in a command that accepts them), issue the command in the following form:

CHDIR

CHDIR reports as follows:

```
C:\KEEP
```

To change the default directory of the disk in drive A: to \FILES when the logged disk is C:, issue the command in the following form:

CHDIR A:\FILES

DOS changes the default directory in A: to \FILES. To confirm the change, issue the command in the following form:

CHDIR A:

CHDIR replies as follows:

```
A:\FILES
```

The subdirectories \WP\MEMOS and \WP\DOCS both have \WP as their parent directory (as their names imply). To change the directory in the current drive C: from \WP\MEMOS to \WP\DOCS, issue the command in the following form:

CHDIR ..\DOCS

Notice that the alias for the parent directory (.. with no leading \) of the current directory acts as a shorthand substitute for the same full command you issue in the following form:

CHDIR \WP\DOCS

To change from the \WP\DOCS directory on the current drive to the \WP (the parent directory of \WP\DOCS), you issue the command in the following form:

CHDIR ..

Reminder:
The root directory has no parent.

DOS will log \WP as the current directory of the logged disk. The use of the .. alias as the only parameter of the command always logs to the parent of the current directory, unless the current directory is the root. The root has no parent.

Defining the Program Search with PATH

The PATH command enables you to define a search path that DOS uses to locate programs if they are not located in your current directory. Once a search path is defined by PATH, the search path remains in effect until you issue the PATH command with another search path.

Executing programs (files with .COM, .EXE, or .BAT extensions) in a single-directory file system is no problem. You simply issue the command or program name, and COMMAND.COM loads and executes it. In a hierarchical directory system, however, especially one that includes a hard disk, your current working directory may not be the directory that contains the entry for the program you want to run. You can include the drive and path specifiers in front of the name of the desired program as part of the command line, but entering the additional specifiers can be a cumbersome process.

DOS provides a alternative to issuing drive and path specifiers in front of a command by enabling you to define alternative search paths. These alternative search paths are used by DOS if the current directory does not yield a desired .COM, .EXE, or .BAT file.

Cue:
The PATH command defines the disks and directories that DOS searches if the command or program is not located in the current directory of the specified (or default) drive's disk.

The PATH command defines the disks and directories that DOS searches if the command or program is not located in the current directory of the specified (or default) drive's disk. Most users provide a PATH command with alternative search paths in their AUTOEXEC.BAT file for the convenience of automatically defining the search paths at the time the computer is booted. The PATH command is also executed at the DOS prompt.

The DOS external commands are programs that have either .EXE or .COM extensions. Because DOS command programs are likely to be needed from any point in the hierarchical directory tree, the external commands are useful in illustrating the efficiencies gained by using the PATH command.

Issuing the PATH Command

The syntax for the PATH command is as follows:

PATH *d1:path1;d2:path2. . .*

Each alternative search path consists of a drive and path pair represented by *dn:pathn*; *n* is the corresponding position of the pair—in this case, 1 or 2.

Issuing PATH alone with no parameters returns a report of the current search path on the display.

The optional drive parameters *d1:* and *d2:* can be any drive names such as A, B, or C. If any of the drive parameters is omitted from the command line, the current drive is searched by default. The same drive can be specified as the drive parameter for different alternate paths. If the specified drive is not a legal drive, DOS will issue the following warning when trying to search the illegal path:

```
Invalid drive in search path
```

If the specified drive is temporarily invalid because of a JOIN or SUBST command, the message can be ignored.

The optional parameters *path1* and *path2* are path names on the corresponding drives. If any of the path parameters is omitted from the command line, the specified disk's root directory is assumed.

The semicolon (;) separates multiple drive and path pairs in the command line. If a semicolon follows the PATH command as the last character of the command line, DOS eliminates the defined search path. No spaces separate drive and path pairs and the semicolon.

The ellipsis (...) indicates that the number of drive and path pairs can vary. You can specify multiple alternative drive and path pairs.

Understanding the Operation of PATH

Normally, the external DOS commands are copied from the DOS master disk to a hard disk (usually C:) directory named \DOS. You can issue one of these commands, for example CHKDSK, from any directory on any disk by including the full path to the command in the following form:

C:\DOS\CHKDSK A:

This command specifies to DOS that the command program, CHKDSK, is located on drive C: in the \DOS directory. Giving command-path specifiers makes commands accessible from any directory, but the extra keystrokes require additional work and provide the potential for typos. External commands issued with command paths do not have the same spontaneity as internal DOS commands in which no characters precede the command name. Issuing COPY as a command name is more spontaneous than issuing C:\DOS\XCOPY. In a DOS work session, you can issue dozens of external DOS commands from various directory locations. You can use the PATH command to eliminate the cumbersome inclusion of the command's path from the command line.

By defining the search path for programs as being C:\DOS, you can ensure that every DOS command will execute from any point in any disk's directory tree. You do not have to think about which command is internal and which command is external. You simply issue the command by its DOS name and save your concentration for providing the correct parameters and switches. If the external command that you use is located in C:\DOS, the PATH definition of C:\DOS provides DOS with an alternative directory to search for the command if the working directory does not include the command. The PATH command is not limited to just one alternative for the current directory. You can define several alternative paths; DOS searches the alternatives in the order in which the paths are defined.

Reminder: *You can define several alternative paths with the PATH command; DOS searches the alternatives in the order in which the paths are defined.*

Understanding the General Rules of PATH

- When alternative search paths are defined by PATH, the alternatives remain in effect until another PATH command modifies the current definition.
- When you turn off the computer or use the Ctrl-Alt-Del sequence to "warm boot" the computer, you must reestablish the defined search path. (Many AUTOEXEC.BAT files set the search path automatically.)

Reminder: *When you reboot the computer, you must reestablish the defined search path unless the path is set in your AUTOEXEC.BAT file.*

- If directories defined by the PATH command do not exist, the nonexistent path(s) are ignored when the alternative search paths are searched. DOS does not issue any message or warning about nonexistent search-path alternatives.
- DOS uses the search paths defined by PATH in the order the alternatives are defined.
- The search-path string (the parameter's characters) defined by the PATH command is passed to executing programs as part of the environment (see the SET command in chapter 8). If the semicolon (;) character follows the last drive and path as the last character of the command line, the semicolon will be the last character of the PATH environment string.

  NOTE: DOS processes the search path as though the final semicolon was not in the search path string. Rarely does a trailing semicolon in the path string present a problem. Certain applications programs that use the PATH environment string, however, may not expect the semicolon and may fail to test the current end of the path line in the AUTOEXEC.BAT file. These programs may fail to process the string.
- To remove the search path, issue the PATH command followed only by a semicolon (**PATH;**).

Using PATH To Search Directories

To establish a search path to the \DOS directory on drive C:, enter the following command:

PATH C:\DOS

You can confirm the new search path by entering **PATH**. DOS then reports the following:

```
PATH=C:\DOS
```

You can confirm that the new search path is in the environment by using the SET command. DOS reports as follows:

```
COMSPEC=C:\COMMAND.COM
PROMPT=$P$G
PATH=C:\DOS
```

You can use the equal sign (=) to separate the command from the parameters. DOS accepts the following command:

PATH=C:\DOS

Confirm the path with the following command:

PATH

The next DOS report shows the acceptance of the equal sign:

```
PATH=C:\DOS
```

To include the root directory of drive C: as the first alternative in the search path, reenter the search path by using the following command:

PATH C:\;C:\DOS

PATH cannot add a search-path alternative to the existing search path by using the contents of PATH as a parameter for concatenation. Because PATH does not confirm paths specified in the command line, you can actually use an erroneous command like the following:

PATH=PATH+C:\DBASE

You can confirm that the new search path, although meaningless, is literally as you specified it by using the PATH command alone. DOS then responds as follows:

```
PATH=PATH+C:\DBASE
```

DOS did not expand the value of PATH into the literal search path of current alternatives plus new alternative, nor did DOS concatenate the existing path and the new path into a resulting search path. This search path is worse than none at all.

To remove a search path, enter the PATH command with a semicolon:

PATH;

To confirm that no search path is in effect, enter just **PATH**; DOS responds as follows:

```
No Path
```

When the search-path parameter ends with a semicolon, as in **PATH C:\DOS;**, the value of the search-path string includes the semicolon. You can confirm this by using the PROMPT or SET command. If you issue the SET command, DOS reports as follows:

```
COMSPEC=C:\COMMAND.COM
PROMPT=$P$G
PATH=C:\DOS;
```

Using Aliases with PATH

Remember that DOS accepts the (..) directory name as an alias for the parent directory of the current directory. You can include the parent alias in the PATH command as a path parameter; DOS searches the current directory's parent directory if the program you want to run is not in the current directory. The logic behind including aliases in the PATH command is not as clear as the logic of including specific directories. The logic of aliases as PATH parameters is based on how many of the applications programs' directories are nested.

Reminder: *Many applications programs create subdirectories during installation.*

Many applications programs create subdirectories when you install them on your hard disk. dBASE IV is a good example. dBASE IV creates its own directory named \DBASE. In addition, it creates a subdirectory named \DBASE\SAMPLES. The installation program copies the DBASE program and associated overlay files (necessary to run DBASE IV) to the \DBASE directory. The installation program copies sample source program files to the \DBASE\SAMPLES directory. Finally, the installation program modifies the AUTOEXEC.BAT command to add the \DBASE directory to the PROMPT command as an additional alternative search path.

When you want to use the sample programs in the \DBASE\SAMPLES directory, you can use CHDIR to change to \DBASE\SAMPLES and still start the dBASE IV program by typing the program name, **DBASE**, at the DOS prompt. DOS fails, however, to find DBASE.EXE in the current directory, \DBASE\SAMPLES, and uses the search path to search for the DBASE.EXE program file in the alternative directories. When the \DBASE directory is searched, DOS finds DBASE.EXE and then loads and executes the program. Because the current directory is still \DBASE\SAMPLES, DOS can find all the sample files in the default directory. By putting its own directory into the search path, dBASE

IV ensures that DOS will find the DBASE.EXE program when the \DBASE\SAMPLES directory is the current directory. Having a search path to \DBASE, however, can affect the efficiency of DOS.

If several of your applications programs include their default directories in the search path, the number of directories and directory entries within each directory can become quite large. DOS is willing to search a large number of directory entries in a search path, but the search efficiency is reduced. DOS might have to wade through hundreds of directory entries before locating the desired program. In the case of a command-line typo such as entering LABLE for LABEL, DOS will search every directory entry in the search path before giving up with the `Bad command or filename` message. Because most applications programs that create their own subdirectories are never more than one or two levels away from their subdirectories, you can improve efficiency by placing an aliased relative path into the search conducted for programs.

You can, for example, issue the PATH command in the following form:

PATH ..;..\..;C:\;C:\DOS;\C:\DBASE

The first two directories in line for a search will always always be the parent of the current directory—and the parent of the parent of the current directory (the grandparent directory). In this example, DOS can find DBASE.EXE from any subdirectory of \DBASE and even from any subdirectory of a \DBASE subdirectory. If you want to start the dBASE program from a point not included in a branch of the \DBASE directory, the search path will fall through to the \DBASE directory after all. Most of the time, however, the aliases in the search path will find the DBASE.EXE program. The same search path benefits other programs too.

With the same search path, you can start the program Reflex from the \REFLEX\DATA\Y_1987 directory and not even need the \REFLEX directory in the search path. DOS finds the REFLEX.EXE program because the search path is relative and backward, reaching from the current directory.

Using aliases as parameters for a PATH command, however, may not be for you. You may simply need literal search paths in your PATH command. Using aliases with PATH, however, is a good example of how these .. directory entries can be used to improve your efficiency with the PC.

Simplifying Data-File Searching with APPEND

The external command APPEND is a program that gives users of DOS V3.2 and later versions an alternative path to data files in a way similar to PATH's alternative path to program files. APPEND is a memory-resident program that loads when you issue the APPEND command; it remains in memory until you reboot.

Reminder: *APPEND is a memory-resident program.*

APPEND offers a solution for programs that expect their data files to be in a specific directory or on a specific disk, when in fact, the data files are located elsewhere. PATH can automatically load and execute program files—those with .COM, .EXE, and .BAT extensions—in the search path. PATH cannot, however, help with programs that need data files or those programs that know nothing about hierarchical directories.

Most software publishers write programs that take advantage of hierarchical directories, making high capacity floppy disks and hard disks easier to use. But some older applications packages that do not use hierarchical directories need special help. If you are using a program that does not take advantage of hierarchical directories, you must keep your programs and data files together in a single directory. You cannot place your programs in one directory and your data files in a subdirectory. If you are like most hard disk users, you have some programs in one directory and the program's associated data in a different directory.

You will find that the most problematic software is the type that requires additional program files while the main program is running. WordStar Release 3 is an example of this type of software. This version of WordStar uses two overlay files, WSOVLY1.OVR and WSMSGS.OVR, while WordStar is running. Although you can put the main WordStar program file in the subdirectory used by the PATH command, you must also keep copies of the two overlay files in each subdirectory you use with WordStar Release 3. WordStar Release 3 users can use APPEND to establish a search path to the OVR files. With the APPEND path, you do not need multiple copies of the overlay files to run WordStar Release 3 from multiple subdirectories. (Fortunately, Releases 4 and 5 of WordStar incorporate hierarchical directories.)

APPEND can therefore be considered the the PATH command for use with data files. The command's operation was changed slightly for DOS V4. If you find it necessary to trick main programs into finding other needed files, you can now use APPEND. APPEND is a more complex command than PATH, and its syntax requirements are somewhat more complicated.

Issuing the APPEND Command

APPEND was introduced as a standard MS-DOS command in MS-DOS V3.2. No switches are available with V3.2 APPEND.

APPEND V3.3 has two switches which specify optional features of the command. The */E* switch tells APPEND to store the APPEND paths as a string in the *environment*. You can then use the SET command to change the value of the APPEND variable to cause a different APPEND search path. The */X* switch, the *extended* search switch, tells APPEND to search not only for data files, but

to search also for program files. The /X switch allows APPEND to search for .EXE, .COM, and .BAT files, much like the PATH command. APPEND must make special provisions in the DOS kernel, however, to access the environment and the program files.

To use APPEND V3.3 with either of these features, you must first issue APPEND with the switch or switches you want to use, but with no path parameters. APPEND then establishes in the DOS kernel the provisions necessary to use the features and thereafter stays in memory. If you issue an APPEND command again, APPEND behaves like an internal DOS command. After the first issuance of APPEND with switches, you can issue subsequent APPEND commands with path specifiers. In effect, establishing an APPEND search path with optional features is a two-command operation.

Cue:
Establishing an APPEND search path with optional features is a two-command operation.

In DOS V4, APPEND includes the */E* and */X* switches as well as a */PATH* switch. V4 APPEND must be issued first without an APPEND path specifier, but with an /E switch so that APPEND can use the environment feature. V4 accepts the /X switch in the first issuance of APPEND. The /X and the /PATH switches may be specified in subsequent issuances of APPEND.

If you do not intend to use APPEND's optional features, you issue the APPEND command with path specifiers the first time. Once APPEND is loaded into memory, you will have to reboot your PC if you change your mind about using the /E and V3.3's /X switches. After rebooting, you can issue your preferred version of the command.

The various forms of APPEND's syntax are listed in the following sections.

Reminder:
The APPEND command has different syntax forms, depending on the DOS version and the use of optional switches.

Issuing the V3.2 APPEND Command

If your version of DOS V3.2 includes the APPEND command, use the command with the following syntax form:

APPEND *d1:path1\;d2:path2\. . .*

d1: is the optional drive specifier of the first alternative of the APPEND search path. If the drive specifier is omitted, the default drive is assumed.

path1 is the optional path specifier of the first alternative of the APPEND search path. If the path specifier is omitted, the root directory is assumed.

; is the alternative APPEND search-path separator. A single *;* parameter with APPEND cancels the current APPEND search path.

d2: and *path2* are optional, additional APPEND search-path alternative specifiers. If additional alternative search paths are specified with APPEND, the additional paths will be searched after the first path alternative. Additional search-path alternatives follow the same syntax rules as the first search-path alternative.

APPEND issued with no parameters reports the current value of the APPEND search path string. Issuing the SET command will report the values of the environment variables—including the current APPEND value if the /E switch was first used with APPEND. V3.2 has no provision for placing the APPEND search-path string in the environment.

Issuing the V3.3 APPEND Command

If you use the APPEND command with V3.3, you must decide before the first issuance of APPEND whether you want the features provided by the */X* and/or */E* switches. If you want the features of either switch, you must first issue the APPEND command in the following syntax form:

APPEND */X /E*

/X is the optional *extended* search switch that enables APPEND to search for and execute programs in a manner similar to the PATH command. If the /X switch is omitted from the command line, APPEND searches only the alternative paths for nonprogram files.

/E is the optional *environment* switch that enables APPEND settings to be updated in the environment. When an APPEND search path is subsequently given in an APPEND command, the search specification is placed into the environment in the APPEND string. The APPEND search path can be modified by setting the APPEND environment string to a new search path value with the following directive:

SET APPEND=*new value* **directive**

Once APPEND V3.3 has been issued with the switches you want, or if you want to use APPEND without switches (and therefore without additional features), you issue the command in the following form:

APPEND *d1:path1\;d2:path2\. . .*

d1: is the optional drive specifier of the first alternative of the APPEND search path. If the drive specifier is omitted, the default drive is assumed.

path1 is the optional path specifier of the first alternative of the APPEND search path. If the path specifier is omitted, the root directory is assumed.

; is the alternative APPEND search-path separator. A single *;* parameter with APPEND cancels the current APPEND search path as in the command **APPEND ;**.

d2: and *path2* are optional, additional APPEND search-path alternative specifiers. If additional alternative search paths are specified with APPEND, the additional paths will be searched after the first path alternative. Additional search-path alternatives follow the same syntax rules as the first search-path alternative.

Issuing the V4 APPEND Command

If you use the APPEND command with V4, you must decide before the first issuance of APPEND whether you want the environment feature provided by the /E switch. If you want APPEND to update the environment, you must issue the first form of APPEND syntax.

APPEND V4 includes the feature offered by the /X switch automatically, although the feature is off by default. You can start APPEND with the extended search capability on by issuing the APPEND command in any form and including the /X switch. If you want the features of /E, and optionally /X on, you must first issue the APPEND command in the following syntax form:

APPEND */X /E*

/X is the optional *extended* search switch that allows APPEND to search for and execute programs in a manner similar to the PATH search capability. If the /X switch is omitted from the command line, APPEND searches only the alternative paths for nonprogram files. In V4, omitting the /X switch from the initial issuance of APPEND does preclude the use of /X in subsequent APPEND commands.

/E is the optional *environment* switch that allows APPEND settings to be updated in the environment. When an APPEND search path is subsequently given in an APPEND command, the search specification is placed into the environment in the APPEND string. The APPEND search path can be modified by setting the APPEND environment string to a new search path value with the following directive:

SET APPEND=*new value* **directive**

If you omit the /E switch from the initial APPEND line command line, APPEND will initialize without the /E feature. To subsequently use the /E feature, you must reboot.

Once APPEND V4 has been issued with the /E switch, or if you want to use APPEND without the /E switch (and therefore without the environment feature), you issue the command in the following form:

APPEND *d1:path1\;d2:path2\. . . /X /X:ON /X:OFF*
/PATH:ON /PATH:OFF

d1: is the optional drive specifier of the first alternative of the APPEND search path. If the drive specifier is omitted, the default drive is assumed.

path1 is the optional path specifier of the first alternative of the APPEND search path. If the path specifier is omitted, the root directory is assumed.

; is the alternative APPEND search-path separator. A single *;* parameter with APPEND cancels the current APPEND search path as in **APPEND ;**.

d2: and *path2*\ are optional, additional APPEND search-path alternative specifiers. If additional alternative search paths are specified with APPEND, the additional paths will be searched after the first path alternative. Additional search-path alternatives follow the same syntax rules as the first search-path alternative.

/X is the optional *extended* search switch that allows APPEND to find and start programs similar to the PATH command. */X* has the same meaning as */X:ON*.

/X:ON is the alternative form of */X* for enabling the extended search feature. */X:ON* is the same as */X* on the command line.

/X:OFF is an alternative of the */X* switch for disabling the extended search feature. When */X:OFF* is in the command line, no PATH-like processing of program files is performed by APPEND.

/PATH:ON is the optional path processing switch that allows APPEND to search the APPEND search path for a file given with a drive and/or full path name. If the file is not in the given full path name or on the given drive, */PATH:ON* allows APPEND to search the APPEND path to find the file. */PATH:ON* is the default state of V4 when APPEND starts with no */PATH* switch in the command line. The current APPEND search path setting is not affected by the */PATH:ON* switch.

/PATH:OFF disables APPEND's ability to search the APPEND path for a file given with a drive or directory specifier. If a given file is not found in the file's given directory or on the given drive, APPEND will not process the file using the APPEND path.

Understanding the Operation of APPEND

Although V3.2, V3.3, and V4 implement the APPEND command syntax differently, the operation of the APPEND command is very similar for all three DOS versions that include the command. The optional switches of V3.3 and 4 modify somewhat the operation of APPEND. The principle of the APPEND search path, however, is the same in all three versions.

When APPEND loads into memory, it modifies the internal DOS table that indicates the location of DOS kernel service routines. The service routines that locate files have entries in this table. APPEND provides file-location service beyond the standard service offered by the DOS kernel routines. APPEND, when loaded, changes the internal table so that the APPEND routine is listed in the table in place of the standard DOS routines.

When a program or command calls on a DOS file-service routine, the APPEND routine is activated in the file-service routine's place. The program is not aware that APPEND is providing the service in place of the standard routine. The pro-

gram "thinks" that the table entry that it called is the standard routine. APPEND is, in effect, "intercepting" a program's request for some standard DOS-kernel file services.

Because APPEND adjusts this internal services table, APPEND can access the original internal table values of the file service without the help of the table. APPEND accepts a program's file-service request and enables the standard file-service routine to process the request first.

Normally, the file-service request to find a file will fail when the file is not in the directory of the disk specified. Because APPEND starts the DOS kernel file-service routine for the program, APPEND can detect when the requested file has not been located. When the standard routine fails, APPEND looks in the first alternative of the APPEND search path for the file. If the first alternative path fails, APPEND tries the next. APPEND tries each alternative search path in turn until the file is found, or until no other alternative path is left. If the file is located by APPEND, the file is turned over to the calling program as though the program's original request for the file were successful. If APPEND can't locate the requested file, APPEND returns a "failed" code to the requesting program as though the program's original file service-request had failed to find the file.

If your program writes information to files, be cautious in using APPEND. Programs without data-file directory specifiers always write information to files in the current directory, regardless of where the original files may be. For example, if the current directory is \WORDS, and APPEND has tricked a program into finding the file RALEY.TXT in the directory \WORDS\CONTRACTS, the program will save the file in \WORDS. The original file in \WORDS\CONTRACT, however, remains unchanged. You can be tripped up when a program saves files in the current directory, and you are not aware of it. Your word processor might find the old version of the RALEY.TXT file when you've requested a printout.

If you misspell a path that you use with APPEND, or if you delete the directory, APPEND does not notice the error until DOS attempts to use the paths. You will not see any error messages if you specify invalid paths. DOS simply skips any missing or misspelled path names and proceeds to the next path. If you use an invalid path name, APPEND may skip the directory you intended APPEND to use.

Understanding the General Rules of APPEND

- V3.2, V3.3, and V4 implement APPEND in slightly different ways. You can use V3.2 syntax with V3.3 and V4. V3.3 syntax can be used with V4.
- APPEND is initially an external command. After you issue the first APPEND command, it acts as an internal command.

- If you want to include the features of the /X and /E switches with V3.3 APPEND, you must issue **APPEND /X /E** before any other APPEND command.
- If you want to include the feature of /E with V4 APPEND, you must issue **APPEND /E** before any other APPEND command.
- The command **APPEND ;** cancels the APPEND path, but APPEND remains memory resident.
- Although APPEND locates a data file in an alternative path, programs that use the data file, modify the file, and write the modified file back to disk will not know where APPEND found the file. The programs will write the file to the current disk and directory or a program-specified disk and directory.
- The V4 /X switch has the same meaning to APPEND as /X:ON. Once the /X or /X:ON switch is in effect, /X:ON remains in effect until an APPEND command with an /X:OFF command is issued.
- The V4 /PATH:ON switch is the default. The /PATH:ON switch remains in effect until an APPEND command with a /PATH:OFF switch is issued.
- The /X switch allows APPEND to locate files with .COM, .EXE, and .BAT extensions.
- The /X switch allows APPEND to work in a similar manner to the PATH command. PATH, however, does not search for a file if the file is specified with a drive or a path specifier. **APPEND /X** will search for the file unless V4 /PATH:OFF is in effect.
- The purpose of APPEND is to allow programs to locate files when the file specification is insufficient to allow DOS to locate the file. Some programs may behave erratically when the program is not designed to anticipate APPEND's effect.

Caution: *Use the APPEND command before you use the ASSIGN command.*

- APPEND must be started before ASSIGN is used. ASSIGN "hides" the true nature of the assigned drive from most commands, and APPEND will behave erratically if ASSIGN is issued first.

Using APPEND with the /X Switch

To see how **APPEND /X** can fool you and your programs, here is a sample session demonstrating how you can be tricked. Place a blank, formatted disk in drive A: and invoke **APPEND /X**. Then set APPEND to use a directory. Now, issue DIR to get a directory of the disk. The result is not what you might expect:

```
DIR A:
Volume in drive A is TEST DISK
Directory of  A:\
File not found
DIR C:\DOS\DRIVERS
Volume in drive C is LAP TOP
 Directory of  C:\DOS\DRIVERS

.            <DIR>      01-01-89   2:14p
..           <DIR>      01-01-89   2:14p
ANSI     SYS     1678   3-17-89  12:00p
COUNTRY  SYS    11285   3-17-89  12:00p
DISPLAY  SYS    11290   3-17-89  12:00p
DRIVER   SYS     1196   3-17-89  12:00p
KEYBOARD SYS    19766   3-17-89  12:00p
PRINTER  SYS    13590   3-17-89  12:00p
VDISK    SYS     3455   3-17-89  12:00p
    9 file(s)  5256794 bytes free
```

Next, issue the following commands:

APPEND /X

APPEND C:\DOS\DRIVERS

DIR A:

The result is as follows:

```
Volume in drive A is TEST DISK

Directory of  A:\

             <DIR>      01-01-89   2:14p
..           <DIR>      01-01-89   2:14p
ANSI     SYS     1678   3-17-89  12:00p
COUNTRY  SYS    11285   3-17-89  12:00p
DISPLAY  SYS    11290   3-17-89  12:00p
DRIVER   SYS     1196   3-17-89  12:00p
KEYBOARD SYS    19766   3-17-89  12:00p
PRINTER  SYS    13590   3-17-89  12:00p
VDISK    SYS     3455   3-17-89  12:00p
        9 file(s)  5256794 bytes free

C>
```

Instead of reporting that there were no files to be found on the disk in drive A:, the DIR command shows the C:\DOS\DRIVERS directory. When there are no files on the disk, APPEND tricks DOS into searching the first directory specified to APPEND. You get a false answer from the DIR command. After you disconnect APPEND, DIR will work correctly.

If you use APPEND with the /X switch, some programs may react in unexpected ways. The result will be that a program that has always worked perfectly will mysteriously not perform correctly. If you encounter this problem, restart DOS and use the APPEND command without the /X switch (or, if you have DOS V4, turn off the feature by typing **APPEND /X:OFF**). If the program then works correctly, remember not to use APPEND with the /X switch with that program. Or restart the computer just before you need to use that program (or, if you have DOS V4, turn APPEND off with a command).

If the program continues to work incorrectly, try using the command **APPEND;** to deactivate APPEND. If the program still does not work correctly, mixing the program with APPEND is not the problem. Although you may not have a problem using APPEND with applications programs, be suspicious if some applications programs do not work correctly with APPEND in effect.

Finally, do not mix DOS V3.3's or V4's APPEND with another version of APPEND, such as the ones provided with the IBM PC Network or IBM PC LAN software. The programs are incompatible, and you will have unsatisfactory results.

Despite these few exceptions, APPEND, like PATH, is a very useful command. With APPEND, you can place your programs in one directory and organize your data files in a subdirectory. What PATH does for command files, APPEND does for data files.

Listing Directories with TREE

The files in your hierarchical directories are probably organized in a manner similar to the way you would physically organize floppy disks. Perhaps you keep subdirectories for letters, for applications programs, for memos, or many other categories. As the number of directories and files grows, however, your ability to remember which directory holds which file decreases. Instead of asking yourself which disk holds your file, your question becomes "Which directory holds my file?"

Cue:
The TREE command lists the directories, and optionally, the files on a disk.

DOS provides an external command—TREE—that lists all the directories of a disk. TREE will also list the files in the directories. If you have a hard disk computer, you will find the TREE command especially useful.

Issuing the TREE Command

TREE's command syntax is as follows:

TREE *d:path*\ */F*/*A*

d: is the optional name of the disk drive that holds the directories you want to list. If you omit *d:*, TREE lists the directories on the current disk drive.

TREE V3.3 and earlier versions accept only the single switch */F*, which directs TREE to list the *files* in the directories. Use this switch if you are trying to locate a file. If you do not give the switch, TREE displays only a list of the disk's directories—not the disk's files.

TREE with MS-DOS 3.3 and PC DOS V4.0 accepts the optional *path*\ specifier. The path specifier names the directory where TREE starts processing the listing. If the path specifier is omitted, TREE begins processing from the root directory.

In V4, TREE's output displays the directory tree with line graphics characters. The */A* switch available with V4 enables you to use an alternate graphics character set.

Understanding the Operation of TREE

TREE is not the same as the DIR command; you do not give a path or file name with the drive name. The TREE command—available with PC DOS through V3.3 and MS-DOS through V3.2—automatically starts its listing with the root directory of the specified or current disk drive; you do not give TREE the choice of where it starts. Also, you cannot specify a file for which you want to search. With TREE, you get an all-or-nothing listing of the disk's hierarchical directory specification.

MS-DOS V3.3 and DOS V4 accept an optional path specifier which causes TREE to list the directory structure for the specified directory and lower directories. With certain implementations of MS-DOS V3.3 and DOS V4, the /F switch displays the file names of each directory in a wide format next to each directory (see fig. 6.2). This feature compresses TREE's lengthy /F report format.

The MS-DOS V3.3 and DOS V4 TREE listing uses line-drawing characters to draw the tree structure (see fig. 6.3). With V4, if you specify the /A switch, TREE displays line-drawing characters which consist of normal keyboard characters (see fig. 6.4). Although the characters produced by the /A switch are not as well suited to drawing lines as TREE's default characters, the characters produced by /A can be printed on any printer. If your printer does not follow the IBM printer character set for extended (ASCII 128 or greater) characters, use the command **TREE /A >PRN** to direct the output to your printer.

Fig. 6.2

The output of ***TREE /F*** *showing listed files.*

```
C:\>TREE \DOS /F

Directory path listing for volume : HARD DISK

\DOS
  4201.CPI       5202.CPI       EGA.CPI        LCD.CPI        APPEND.EXE
  ASSIGN.COM     ATTRIB.EXE     BACKUP.COM     CHKDSK.COM     COMP.COM
  DEBUG.COM      DISKCOMP.COM   DISKCOPY.COM   EDLIN.COM      FASTOPEN.EXE
  FDISK.COM      FIND.EXE       FORMAT.COM     GRAFTABL.COM   GRAPHICS.COM
  JOIN.EXE       KEYB.COM       LABEL.COM      MODE.COM       MORE.COM
  NLSFUNC.EXE    NOW.EXE        PRINT.COM      RECOVER.COM    REPLACE.EXE
  RESTORE.COM    SELECT.COM     SHARE.EXE      SORT.EXE       SUBST.EXE
  SYS.COM        TREE.COM       XCOPY.EXE      TEST1.EXE      SETUP1.EXE
  CHAD.COM       EXE2BIN.EXE    FC.EXE         LINK.EXE       README.TXT

  \DRIVERS
    ANSI.SYS       COUNTRY.SYS    DISPLAY.SYS    DRIVER.SYS     EMM.SYS
    KEYBOARD.SYS   PRINTER.SYS    VDISK.SYS      EGA32CPY.SYS

C:\>
```

Fig. 6.3

The TREE command showing line graphics characters.

```
D:\>TREE
Directory PATH listing for Volume LOGICAL_D
D:.
├───123BB3
│   └───ALLWAYS
├───DBASE3
├───PM
│   ├───GETSTART
│   └───TEMPLATE
├───DBASE4
│   ├───SQLHOME
│   ├───TEMPLANG
│   └───UTIL
├───SOURCE
├───DSWORK
├───ANSWERS
├───BGI
├───EXAMPLES
├───PIE
├───PROGRAMS
└───REFEXAMP

D:\>
```

```
D:\>TREE /A
Directory PATH listing for Volume LOGICAL_D
D:.
+---123BB3
!   \---ALLWAYS
+---DBASE3
+---PM
!   +---GETSTART
!   \---TEMPLATE
+---DBASE4
!   +---SQLHOME
!   +---TEMPLANG
!   \---UTIL
+---SOURCE
+---DSWORK
+---ANSWERS
+---BGI
+---EXAMPLES
+---PIE
+---PROGRAMS
\---REFEXAMP

D:\>
```

Fig. 6.4

The output of ***TREE /A*** *showing alternate graphics characters.*

Using TREE To View a Disk's Hierarchical Structure

The following examples show the TREE command used on a practice disk. The example provides a display similar to that found with PC DOS V3.3. For DOS V4, TREE's output uses block graphics and indenting to show the hierarchical nature of the directory structure a bit more clearly.

The first example shows the output of the command **TREE A:**

```
C>TREE A:

DIRECTORY PATH LISTING FOR VOLUME PRACTICE

Path: \COMS

Sub-directories:  SAMPLES

Path: \COMS\SAMPLES

Sub-directories:  TEST

Path: \COMS\SAMPLES\TEST

Sub-directories:  None
```

The next example shows the output of the TREE command when the /F switch is included:

```
C>TREE A: /F

DIRECTORY PATH LISTING FOR VOLUME PRACTICE
Files:              None
Path: \COMS
Sub-directories:  SAMPLES
Files:            COMMAND .COM
                  ASSIGN  .COM
                  BACKUP  .COM
                  BASIC   .COM
                  BASICA  .COM
                  CHKDSK  .COM
                  COMP    .COM
                  DEBUG   .COM
                  DISKCOMP.COM
                  DISKCOPY.COM
                  EDLIN   .COM
                  FDISK   .COM
                  FORMAT  .COM
                  GRAFTABL.COM
                  GRAPHICS.COM
                  KEYB    .COM
                  LABEL   .COM
                  MODE    .COM
                  MORE    .COM
                  PRINT   .COM
                  RECOVER .COM
                  RESTORE .COM
```

```
                SELECT   .COM
                SYS      .COM
                TREE     .COM
Path: \COMS\SAMPLES
Sub-directories:  TEST
Files:            MORTGAGE.BAS
Path: \COMS\SAMPLES\TEST
Sub-directories:  None
Files:            BASICA   .COM
```

Comparing CHKDSK and TREE

Remember that CHKDSK can be used to produce a report similar to TREE (see Chapter 5). The CHKDSK command accepts the /V switch, which in this case means give a *verbose* listing of the disk's directories and files. The command syntax for the TREE-like version of CHKDSK is as follows:

Reminder: *The command* ***CHKDSK /V*** *produces a listing similar to that of TREE.*

CHKDSK *d: /V*

CHKDSK /V is like the TREE command. *d:* is the optional name of the disk drive to be checked. The /V switch lists the directories as well as the files. The following sample output shows the results when the CHKDSK command is used on a practice disk:

```
CHKDSK A: /V
Volume PRACTICE    created Aug 24, 1989 10:52p
     Directory A:\
     A:\PRACTICE
     Directory A:\COMS
     Directory A:\COMS\SAMPLES
     A:\COMS\SAMPLES\MORTGAGE.BAS
Directory A:\COMS\SAMPLES\TEST
A:\COMS\SAMPLES\TEST\BASICA.COM
A:\COMS\COMMAND.COM
A:\COMS\ASSIGN.COM
A:\COMS\BACKUP.COM
A:\COMS\BASIC.COM
```

```
A:\COMS\BASICA.COM
A:\COMS\CHKDSK.COM
A:\COMS\COMP.COM
A:\COMS\DEBUG.COM
A:\COMS\DISKCOMP.COM
A:\COMS\DISKCOPY.COM
A:\COMS\EDLIN.COM
A:\COMS\FDISK.COM
A:\COMS\FORMAT.COM
A:\COMS\GRAFTABL.COM
A:\COMS\GRAPHICS.COM
A:\COMS\KEYB.COM
A:\COMS\LABEL.COM
A:\COMS\MODE.COM
A:\COMS\MORE.COM
A:\COMS\PRINT.COM
A:\COMS\RECOVER.COM
A:\COMS\RESTORE.COM
A:\COMS\SELECT.COM
A:\COMS\SYS.COM
A:\COMS\TREE.COM
  730112 bytes total disk space
       0 bytes in 1 hidden files
    3072 bytes in 3 directories
  361472 bytes in 27 user files
  365568 bytes available on disk
  654336 bytes total memory
  522368 bytes free
```

Notice that CHKDSK performs its normal disk analysis in addition to listing the files and directories.

Controlling the Output of TREE with Filters and Redirection

With TREE (and CHKDSK /V), the resulting output can be lengthy. This type of output is ideally suited for redirection. You can, for example, use the MORE filter to display one screen's worth of information at a time. The command syntax for using MORE for output redirection with TREE is the following:

TREE | MORE

You can redirect the output of TREE to the printer with this command:

TREE > PRN

You can place TREE's output into a disk file with this command:

TREE > treelist.txt

If you store the output in a disk file, you can print the output at any time. It's a good idea to occasionally run TREE on your hard disk to get a printed copy of your disk's entire contents, which you can keep near your computer. This type of output is a good road map of your hard disk's structure and can help co-workers or friends when they must use your computer.

Reminder: *Redirecting the output of TREE to a disk file or to the printer can provide a handy road map of your directory structure.*

Combining Copying and Directory Management with XCOPY

The external command XCOPY is used to copy files. XCOPY also has the capability to create directories. Because XCOPY can create directories, the command is included in this chapter with the other directory-management commands.

Reminder: *The XCOPY command can copy files and create directories.*

The XCOPY command, introduced with DOS V3.2, addresses the needs of three principal types of computer users: those who have more than one computer, those who have hard disks, and those who need more selectivity than the standard COPY command offers. Almost all PC users fit one or more of these categories, so XCOPY is an important command to know and use.

Issuing the XCOPY Command

XCOPY's syntax is similar to COPY's syntax, but the switches are more complex. XCOPY's syntax is as follows:

XCOPY *ds:paths\filenames.exts*
dd:pathd\filenamed.extd /V /P /W /S /E /A /M /D:date

The *source* files, designated by an *s*, are the files to be copied. If you omit the disk drive name (*ds:*), the current disk drive is used. If you omit the path name (*paths*), the current directory on the drive is used. You can use wild cards in the file name (*filenames.exts*). If you omit the file name, XCOPY assumes that you want to specify a wild card *.*, and all files in the given path will be copied. If you specify no path and file name, XCOPY issues an error message.

With XCOPY, the *destination* files are designated by the *d*. File names, disk drive names, and path names follow the *rule of currents*. This means that if you do not specify a drive name (*dd:*), XCOPY uses the current disk drive. If you do not specify a path name (*pathd*), XCOPY uses the current disk's directory. If you omit the file name (*filenamed.extd*), the copied files retain their previous names.

Cue: *With the XCOPY command, file names, disk drive names, and path names follow the rule of currents.*

If you give a destination name that might be a path name or a file name, and the potential path does not exist on the destination, XCOPY asks you whether the destination is a file name or path name.

As an example, look at the following command:

XCOPY C:\WORDS*.* A:\WORDS

If you use this command to copy the files from C:\WORDS to the disk in drive A:, and the disk does not have a directory called WORDS, XCOPY will display this message:

```
Does WORDS specify a filename
or directory name on the target
(F = file,D = directory)?
```

If the destination is a file name, answer F. If the destination is a directory, answer D. Unlike COPY, XCOPY creates directories on the destination disk as needed.

Understanding XCOPY's Switches

XCOPY has eight switches, which are described in this section. A complete table of XCOPY's switches can be found in the Command Reference section.

/V is the familiar *verify* switch. XCOPY verifies that it has copied the files correctly.

Reminder: *XCOPY is an external program that must be loaded from a disk.*

/W causes XCOPY to prompt and *wait* for you to insert the source disk. This switch is particularly important to users of floppy disk drives. XCOPY is an external program that must be loaded from a disk; the copying process normally starts after the program loads. If you want to copy files between disks—none of which have the XCOPY program—use the /W switch. After XCOPY starts, the program prompts you to insert the correct disk. Remove the disk that holds XCOPY, insert the correct disk, and press any key to start the actual copying process.

/P is the *pause*-and-ask switch. XCOPY displays the name of the file it will copy and asks whether the file should be copied. For the file /MYFILE.123, XCOPY will display the following prompt:

```
/MYFILE.123 (Y/N)?
```

Answer Y to copy the file or N to skip it. When you answer Y, XCOPY immediately copies the file.

Reminder: *XCOPY can traverse the subdirectory tree and copy complete directory branches from one disk to another.*

The */S* and */E* switches affect how XCOPY handles additional subdirectories. These two switches show the true power of XCOPY. COPY limits itself to handling the files from one directory; XCOPY starts with the named or current

directory and can process the files in all additional subdirectories of the directory, all lower subdirectories of these subdirectories, and so on. XCOPY traverses the subdirectory tree and can copy complete directory branches from one disk to another.

The /S switch affects the source and destination directories. The /S switch instructs XCOPY to copy all designated files from the current and subsequent *subdirectories* to a parallel set of subdirectories on the destination disk drive. As an example, look at the following command:

XCOPY C:\WORDS*.* A: /S

The command executes in the following sequence:

1. XCOPY copies all files in \WORDS to the current directory on drive A:.
2. XCOPY copies all files from WORDS\LETTERS to a subdirectory of the current directory called \LETTERS on drive A:.
3. XCOPY copies all files from the subdirectory WORDS\CONTRACTS to the subdirectory of the current directory on drive A: called \CONTRACTS.

In essence, XCOPY lifts a copy of the subdirectory tree starting at the source directory and transplants the copy to drive A:.

Note that XCOPY does not place the copied files from the subdirectories into a single directory. XCOPY places files from one subdirectory into a parallel subdirectory. If the subdirectory does not exist on the destination, XCOPY creates the subdirectory.

The */E* switch affects the operation of the /S switch. If XCOPY encounters an *empty* subdirectory on the source drive, XCOPY /S skips the empty subdirectory. If you give the /S and /E switches, XCOPY also creates empty subdirectories on the destination drive. If you give the /E switch without giving the /S switch, the /E switch has no meaning. You must also use /S to get any action from /E.

The switches */M*, */A*, and */D:date* control which files that match the source file name will be copied. The */M* switch tells XCOPY to copy any file that has been *modified*. /M tells XCOPY to copy a file you have not previously backed up or copied with XCOPY.

In Chapter 7, you will learn more about the archive attribute that is stored in the directory with each file name. When you create or change a file, this attribute is turned on. XCOPY checks to see whether the archive attribute has been set. The set (on) archive attribute will show as an A to the left of the file name reported by the ATTRIBUTE command. DOS turns the archive attribute on when a file is created or modified. When you give the /M switch, XCOPY processes this file. If the attribute is not set, XCOPY skips the file.

The */A* switch works like the /M switch. XCOPY will process only those files that have the *archive* flag turned on. /M clears the archive attribute after XCOPY has copied a file, however; /A does not. This difference can be important. As you will learn later in this book, the BACKUP command can select files based on the archive attribute. If XCOPY has cleared this flag, however, BACKUP will not process the file. If you therefore use XCOPY /M and BACKUP /M (the switches have identical meaning for the two commands), the backup you make using BACKUP may not be complete. Unless you customarily use XCOPY as a backup program, avoid using the /M switch; use the /A switch instead.

The */D:date* switch selects files based on their *directory date*, which is the date of the file's creation or modification. XCOPY copies files that have been created or modified on or after the date specified.

Understanding the Operation of XCOPY

XCOPY is best described as a hybrid between COPY and BACKUP/RESTORE, the disk-backup commands discussed in Chapter 7. XCOPY and COPY both copy files between disks. Unlike COPY, however, XCOPY does not copy files to a nondisk device, such as the printer (PRN) or the console (CON). Like BACKUP and RESTORE, XCOPY can selectively copy files and traverse the directory tree to copy files from more than one directory. XCOPY can also make a destination directory when the destination directory does not exist.

Like COPY but unlike BACKUP, XCOPY copies files that are directly usable: you cannot use files processed by BACKUP until you have processed them with RESTORE.

Understanding the General Rules of XCOPY

- XCOPY cannot copy hidden source files.
- XCOPY will not overwrite read-only destination files.
- If a file specifier is omitted in the XCOPY syntax, XCOPY assumes the *.* full wild-card pattern as the file specifier.
- If you include the /D switch, the date specifier must be entered in the format of the system's DATE command or in the format indicated by the latest COUNTRY command.
- The /V switch performs the same read-after-write checking as the SET VERIFY ON global verify flag.
- To use XCOPY to copy empty source subdirectories, you must specify both the /S and the /E switches.

Cue:
The command ***XCOPY /V*** *performs the same read-after-write checking as the command* ***SET VERIFY ON***.

Using XCOPY Effectively

Because XCOPY can control which files should be copied by date or archive attribute, can copy complete subdirectory trees, and can confirm which files should be copied, the command has several ideal uses. One use is to copy files selectively between disks or directories. A second major use is as a "quickie" hard-disk backup command if you want to back up only a few critical files in several subdirectories. The third major use is to keep the directories of more than one computer synchronized.

Cue: *XCOPY can be used as a quick hard disk backup command.*

With COPY, your control is limited. COPY copies all files that match the given name, which is an all-or-nothing approach. If you use the /P switch with XCOPY, however, you can select all the files that match the source file name; XCOPY will ask you if each file should be copied. You can give a single "larger" file name (a name that selects more files than necessary but covers all the files you may want to copy); then you can select individually which files you really want to copy.

Cue: *XCOPY can be used to keep the directories of more than one computer synchronized.*

XCOPY is practical to use if you want to make backup copies of less than a diskful of files from several directories. Rather than using BACKUP, you might prefer to use XCOPY /A to select files that have changed since the last backup.

This technique has one drawback: you should ensure that the files will fit on one disk. When the destination disk fills, XCOPY stops. XCOPY cannot gracefully handle full disks. You must change disks and restart the XCOPY process using the /A and /P switches to skip the files you have already copied. Also, XCOPY cannot handle files that occupy more than one disk. If you need to back up such a file, use BACKUP.

Reminder: *XCOPY cannot handle full hard disk backups.*

A favorite use of XCOPY is to keep synchronized the contents of the hard disks of two computers. Many people have one computer at work and another at home. If both computers have hard disks, keeping the copies of programs and data files current is a major task. Which files did you change today? Which machine has the more current version?

In such a situation, you will find especially useful the /A switch or the /D:date switch if you use the /S switch on the source and destination computers. The /S switch forces XCOPY to play a hunting game you otherwise need to perform manually. Which switch you use, /A or /D, depends on how often you copy files between the machines. If you copy files between the machines frequently, you will most likely prefer the /A switch. If you let many days pass between synchronizing your computers' contents, you may find that the /D switch works better. If you run BACKUP on the source machine between the times you use XCOPY, you must use the /D switch with XCOPY. BACKUP resets the archive attribute, and XCOPY will not catch all files that have changed.

Removing an Unneeded Directory with RMDIR (RD)

Reminder:
RMDIR removes directories that are at the end of a path branch of the directory tree.

When you need to remove a directory from your directory tree structure, use the RMDIR or RD command. The two commands are different names for the same command; the action of both names is identical. RMDIR removes directories that are at the end of a path branch of the directory tree. Just as you can create a directory structure with MKDIR, you can remove a directory structure with RMDIR.

Issuing the RMDIR Command

RMDIR and RD are different names for the same command. You can issue the command with either command name. The syntax for RMDIR is as follows:

RMDIR *d:path***dir_name**

d: is the optional drive name that holds the disk where the command will search for directory to be removed. If *d:* is omitted from the command line, DOS assumes that the current (logged) disk holds the directory to be removed.

path\ is the valid, existing path to **dir_name**. If the path specifier is omitted, DOS assumes that **dir_name** is a subdirectory of the current directory of *d:*. If *path* begins with a backslash (\) character, DOS assumes that the path to the directory name is absolute from the root. If the path specifier does not begin with a backslash, DOS assumes that the path to the directory name is relative from the current directory of *d:*.

dir_name is the name of the directory to be removed. The directory name is required in the command line.

Understanding the Operation of RMDIR

Reminder:
DOS will only remove an empty subdirectory—one that contains no files other than the . and .. entries.

Deleting a subdirectory is similar to deleting a user file. Subdirectory entries are special files to DOS. Unlike user files, however, subdirectories may contain user files *and* subdirectory entries. DOS will not "orphan" a subdirectory's files and subordinate subdirectories by simply deleting the subdirectory as it does a user file.

When you issue the RMDIR (RD) command to delete a subdirectory, DOS checks to ensure that the subdirectory is empty. The subdirectory will contain the . and .. file entries, but DOS knows that these two entries are part of the subdirectory bookkeeping function. The presence of only the . and .. files in the

directory tells DOS that the directory is empty. DOS also checks to ensure that the current directory is not the directory you are deleting or one of its subordinate directories.

Hidden files which do not appear in a DIR command listing of the directory are considered user files; the directory cannot be removed if it contains hidden files. The CHKDSK command will show the presence of hidden files on a disk. A directory that contains even one subdirectory is not an empty directory.

Reminder: *Use the CHKDSK command to check for the presence of hidden files in a directory.*

After determining that the subdirectory to be deleted is empty, and not the current directory, DOS marks the subdirectory's parent directory entry with a deletion indicator. The deletion indicator tells DOS that the subdirectory is no longer in the file system. A deleted subdirectory entry's indicator in the parent directory tells DOS that the subdirectory entry can be recycled. The deleted subdirectory's allocated space on the disk as tracked in the FAT is then available for DOS reallocation.

Understanding the General Rules of RMDIR

- The directory you remove must be empty.
- The root directory is not a subdirectory and has no entry in a parent directory. You cannot remove a root directory.
- The number of characters in the full path name (drive, path, and directory) to be removed cannot exceed 63.
- A directory affected by a SUBST command cannot be removed.
- Do not remove directories from a drive which is affected by a JOIN or ASSIGN command.
- The directory to be deleted cannot be the current directory of any disk.

The SUBST, JOIN, and ASSIGN commands are covered in Chapter 9.

Deleting an Empty Directory with RMDIR

When you are maintaining files, you may create a directory to contain files for a special purpose of limited duration. Perhaps you keep in a special directory a few files that need to be checked. After you check the files, you no longer need to keep the files separate, or you may not need the files at all. The directory itself may have no further use to you, and you now want to remove the directory from your disk.

As an example, assume that drive E: of your system includes the directory \PROLOG\DSWORK\DONE, which contains program files which have been checked by their author. The files and the directory are no longer needed, so they can be removed. By logging to drive E: and changing to the \PROLOG\DSWORK directory, you will be logged to the parent directory of \PROLOG\DSWORK\DONE. Because the directories in this example do not use the dot (.) character, you can see the directory to be deleted in its parent directory's listing by using the following command:

DIR *.

DOS reports the directory on-screen as follows:

```
Volume in drive E is LOGICAL E
Directory of  E:\PROLOG\DSWORK

.            <DIR>       7-05-89   11:54a
..           <DIR>       7-05-89   11:54a
DONE         <DIR>       7-11-89    9:42a
       3 File(s)    200704 bytes free
```

The directory to be deleted, DONE, is clearly in a subdirectory of the current directory as evidenced by its listing.

From the parent directory of the directory to be deleted, you can issue the RMDIR command with no drive and path specifiers as in the following:

RMDIR DONE

DOS responds with the following message:

```
Invalid path, not directory,
or directory not empty
```

This DOS message tells you that one of the following situations is the case:

1. The full path name you gave (or that was implied by default) is not on the disk.
2. The full path name you specified names a file, not a directory.
3. The directory you are asking DOS to delete still contains files other than the . and .. entries.

You must determine which message applies to your situation by using the process of elimination.

You give directory parameters in commands by including paths. When most commands with path parameters carry out their actions, the objects of the actions are files. DOS does a good job of hiding from users the randomness of the disk locations of files by keeping hierarchical order in the directories.

By checking the current directory with the CHDIR command, or by looking at the prompt (if you have incorporated a current directory indicating the PROMPT command), you can verify that the path is valid. By looking at the parent directory and seeing that \DONE is listed as a directory, you can be assured that the directory name you are using really is a directory, not a file name. The remaining message indicates that the directory is not empty. To determine whether the DONE directory contains user files or subdirectories, you can change to that directory by using the following command:

CHDIR DONE

Then issue the DIR command for all files. You will see the following directory listing:

```
Volume in drive E is LOGICAL E
Directory of  E:\PROLOG\DSWORK\DONE

.            <DIR>       8-11-89   10:23a
..           <DIR>       8-11-89   10:23a
DWIMDOS PRO      3439    6-03-89    5:23p
DWIM1   PRO      5414    6-04-89   11:07a
DWIM2   PRO     11625    6-06-89    5:39p
        5 File(s)     200704 bytes free
```

The directory listing of the DONE directory shows that the directory is not empty. You can erase all files that contain the .PRO extension to empty the directory; then verify that it is empty by issuing another DIR command. The DIR command will report the following:

```
Volume in drive E is LOGICAL E
Directory of  E:\PROLOG\DSWORK\DONE

.            <DIR>       8-11-89   10:23a
..           <DIR>       8-11-89   10:23a
       2 File(s)     255280 bytes free
```

Now only the . and .. alias directory entries remain. You can change to the parent directory by using the following shorthand command:

CHDIR..

From DONE's parent directory, you can remove the now empty DONE with the following command:

RMDIR DONE

The E:\PROLOG\DSWORK\DONE directory is removed.

Taking Advantage of the Hierarchical Directory System

DOS supplies each disk with a root directory. Beyond the root, you can use the directory-management commands to create a directory tree in any architecture you want. DOS is flexible in allowing you to control the external look of your directory tree, while DOS manages the internal bookkeeping tasks of the DOS hierarchical directory system. Still, some planning on your part will help make your directory system design easy to use. The following sections suggest some directory structures that you may want to consider. If you already have your directories arranged, keep these suggestions in mind for the next time that you reorganize your hard disk.

Keeping a Clean Root

Cue: *Keep your root directory clean by keeping in the root only the few necessary files that DOS needs to get started.*

Because DOS defaults to the root directory of the boot disk, the root is a likely place for you to save some of your files. As long as you observe the limit to the number of root directory entries, DOS will not object. If you have a floppy-only system, you may find that the boot disk leaves little room for additional files and that you normally swap the boot disk for a work disk anyway. If you have a hard disk, consider keeping the root directory as clean as possible. Keep in the root only the few necessary files that DOS needs to get started.

If you are using an AUTOEXEC.BAT file, it must reside in the root directory of the boot disk. The same rule applies to the CONFIG.SYS file, if you include one. If the disk is bootable, the hidden DOS system files will be located in the root also, but the FORMAT command puts them there. The system files will remain in the root unless you use special disk-editing software to change the hidden and system attributes so that you can erase them or move them. Be aware that if you erase or move the system files, DOS will not boot.

COMMAND.COM is normally in the root by virtue of the /S switch of the FORMAT command. You do not have to leave COMMAND.COM in the root, however, if you make some entries in the CONFIG.SYS and AUTOEXEC.BAT files. A good directory for COMMAND.COM is \DOS. In your CONFIG.SYS file, add the following line:

```
SHELL=C:\DOS\COMMAND.COM
```

Caution: *DOS will lock up at boot time if you name an erroneous file in the SHELL directive.*

(See Chapter 14 for more information on CONFIG.SYS.)

When DOS is booted, it takes the assignment of SHELL and uses the path and file name to find the command processor. Be sure not to misspell any parts of the full path name. DOS will lock up at boot time if the assignment to SHELL is

a erroneous file. Of course, a working copy of DOS on a floppy will make any SHELL error (or any other hard disk boot failure) easier to recover from. Once you put this line in CONFIG.SYS, do not reboot until you have actually copied COMMAND.COM to \DOS.

You must enter a line or alter an existing line in your AUTOEXEC.BAT file to change the COMSPEC variable. DOS uses the contents of COMSPEC to find and reload COMMAND.COM after COMMAND.COM is loaded at boot time; subsequently the transient portion of COMMAND.COM is pushed out of working memory by a large program. At boot time, COMSPEC is set to whatever CONFIG.SYS says from the SHELL assignment. In this case, COMSPEC is C:\DOS\COMMAND.COM. In the AUTOEXEC.BAT file, find the COMSPEC line or add it if it doesn't exist yet. The line is as follows:

```
COMSPEC=C:\DOS\COMMAND.COM
```

Again, make sure that COMMAND.COM has been copied from the root to \DOS. When you reboot, DOS finds COMMAND.COM in \DOS, and your boot disk's root is free of the file.

Some users keep device drivers, the programs specific to the operation of input/output devices, in the root directory. If you have these types of files in your root directory, you can move them. The next section discusses driver files.

It is not uncommon to find some batch files in a root directory. Batch files are ideal for relocation from the root because you can put them in another directory and let an alternate search path given in a PATH command lead DOS to them. The next section also gives some considerations for a home for batch files.

When you are copying a floppy disk full of files to your hard disk, it is easy to forget to use CHDIR to change the intended destination directory. If you do not use the full destination path name in the COPY command, but instead rely on the convenience of DOS defaults, you can end up with your root full of files from the floppy. If your root is fairly clean (contains only a few files), you can recover from this kind of goof by copying everything in the root to an empty temporary subdirectory, erasing all files in the root, and copying the root's necessary files back to the root from the temporary directory. You can erase the temporary directory's contents and try your original (and errant) COPY command again, but this time, with the correct destination parameters.

Including a Directory To Store DOS Programs

Cue:
Include a separate directory for the purpose of storing the DOS program files and associated programs.

The DOS external commands and other DOS programs, such as BASICA (GW—BASIC), EDLIN, DEBUG, and setup programs, have one important element in common: these programs are all files that directly relate to the functions of DOS. Including a directory to hold these programs and their associated

files is a good idea. The name for this DOS directory can be \DOS, and as the name implies, the \DOS directory is a subdirectory of the root. Some users follow the UNIX operating system's convention of keeping the utility commands in a directory called \BIN. Either name works from an organizational point of view.

When you issue a PATH command on the command line or in an AUTOEXEC.BAT file, position the \DOS entry right after the root entry. Don't forget to include the drive specifier with each alternative search path in the PATH command line. Because all of your external DOS commands are located in the second alternative search path, you should get a good response when you are working in DOS.

Cue:
Keep your driver programs in a separate directory.

Many users prefer to make a subdirectory of \DOS called \DOS\DRIVERS to keep their DOS-included drivers (those files with the .SYS extensions) as well as drivers included with applications programs or hardware. Because drivers are loaded during the processing of CONFIG.SYS, you can place the full path name to the drivers you need in the CONFIG.SYS file and therefore eliminate the need to include a search path to the drivers with PATH. An example of a CONFIG.SYS line might be as follows:

```
DEVICE=\DOS\DRIVERS\MOUSE.SYS
```

Separating the drivers from the DOS commands can make your tree listings from the TREE command more indicative of each subdirectory's file group's function in the \DOS tree branch.

Batch files are like program files when DOS cannot locate them in the current directory. The search path set in the PATH command will locate and execute desired program and batch files located in one of the alternate search paths. Creating a \DOS\BAT directory for all batch files is a good idea.

The length of the full path (drive, path, and new directory name) must not exceed 63 characters including delimiters. The full path name is the measure for length. If you are creating a directory relative to your current directory, you must count your current directory's full path length from the implied root \ character as being part of the 63 limit.

Cue:
Include a search path in your PATH command for batch files stored in a separate utility directory; you can then execute these commands from anywhere in the directory tree.

The PATH command can include an alternative search path for utility batch files; you can then execute these files from any point in the directory tree.

Ensuring Uniform Directories for Applications Programs

Many of today's applications programs use subdirectories to store sample files, tutorials, auxiliary utilities, and your data for the applications. Most of these subdirectory-capable programs include default directory names that are used

when you install the software. Most programs, however, give you the capability to choose your own names for the subdirectories instead of the default names. In most cases, you can let the package install with the default names for directories. Under one circumstance, however, you may not want to use the default names. You may want to control the names of the package's directories if you have an older version of the package and want to keep the older version. dBASE III Plus and dBASE IV, or Lotus 1-2-3 Release 2.1, 2.2, or 3, are examples of this type of situation.

If you need to keep operating an older version of a software package, for example, to maintain older data or program files, allowing the defaults to decide where the program and associated files will load is an invitation to lost dual functionality. The new version may copy files with the same names as old files into the default directories because the default directories from the new version are named the same as your old directories.

Read the installation information for a package carefully before installing the software. You will not be happy if you "return" your way through the installation default directories and find that you cannot start your old version of dBASE or 1-2-3. The old version can be overwritten in installation. Pick a new set of directory names for the new package, such as \DB4 for dBASE IV or \123R3 for Lotus 1-2-3 Release 3. The contents of the previous versions' directories will be unaltered by the installation process.

If you have an applications package that does not create directories during installation, you should think about how you want to structure your own directories for the package and its data. WordStar Release 4, for example, can be copied to a directory named \WS4. The directory or directories for your document files should be subordinate to the \WS4 directory so that all of WordStar's associated files will form a branch of the directory tree and not be spread across branches of a directory tree. You can create a \WS4\MEMOS directory for your memo documents and a \WS4\FORMS for office-form master documents.

Keeping associated files in the same branch of the directory tree also makes the job of backing up files easier. The command **BACKUP /S** backs up all files from the named directory and its subdirectories. You can make convenient partial interim backups of one branch of the directory tree if all associated files you want to back up are in that branch.

Using a Temporary Directory

On many occasions, a temporary directory comes in handy for holding files that you are readying for a final disposition. An example is the use of the temporary directory for recovering from an errant COPY command that deposited files in the wrong directory. You can create a directory called \TEMP to act as a tem-

porary storage location for such files. A single-floppy disk system is not a good system on which to copy floppy disks. The drive gets the job done, but the COPY command, using the same drive for the source and the target, asks for multiple disk swaps. You can use the DISKCOPY command, which might require fewer swaps, but any fragmentation of the original disk is mirrored on the copy.

If you are making multiple copies of an original, a simple and speedy way to make the copy is to copy the source files to the \TEMP directory. You can then copy the files from the \TEMP directory to the destination floppy in the sole floppy drive. The process reduces disk swapping, and the copies will be finished faster thanks to the hard disk's extra speed. When the copies are finished, you can erase the contents of \TEMP so that it will be ready for your next use.

Keeping Files in \KEEP

Sometime a file is like a favorite old shirt. You just don't know if you want to get rid of it. You leave it in a subdirectory like an old shirt in the corner of your closet. You might need it again, but then again, you might not. In either case, you may be too distracted by other issues to make a decision about its disposition when you stumble across it yet another time. Soon, you will change directories and forget about the file—that is—until your disk space gets short. Then you will be digging through the dark corners of your subdirectory closets trying to find a file or two to discard. And at "disk-space-running-out" time, you feel less equipped to make a decision about its importance than ever before. In confusion, you decide to erase the old file.

If this series of events strikes a familiar chord, you might consider creating a \KEEP directory. When you find a file that has questionable future use, but you don't want to be bothered with making a decision about its disposition at the moment, copy it to \KEEP. As always, make sure the file will not overwrite a different file with the same name. You should always give your files descriptive names. Avoid generic names like TEMP and SAVE. Remember that you can always rename files as a function of COPY.

Erase the copy of the file from the subdirectory where you found it to keep that subdirectory uncluttered. When a week is up, \KEEP may have several files in it.

At this point, you may not have saved any disk space, but you have done some housecleaning in a few directories. Now comes the hard part. Before \KEEP contains more bytes of files than your floppy will hold on one disk, sit down and decide which files stay and which ones go. You will likely copy some back to their original directories. Some you will erase, and some you will copy to a "just-in-case" floppy disk for an archive copy. Before you store the floppy away, issue the following command:

DIR *d:*/**W** >**PRN**

d: is the name of your floppy disk drive. This command with the /W switch will send a wide directory report to your printer. Put the printed copy in the disk's envelope for reference. With the floppy disk holding the \KEEP "keepers," you are free to erase the contents of \KEEP. If you use this method regularly, you will develop a better sense of the files that are important; those that aren't as important, you can erase. The prices of floppy disks are low enough that you can offset the cost of a year's worth of \KEEP files on floppies the first time one of your \KEEP floppies contains that file it took you hours to create.

Summary

This chapter covered the commands which are used to manage the hierarchical directory structure. The hierarchical directory system is an important feature of DOS. Your computing experiences will be more productive when you understand and manage your directory tree effectively.

In this chapter, you learned the following key points:

- ❑ Subdirectories were introduced with DOS V2.0.
- ❑ DOS manages the directory structure internally; you manage the directory structure externally with commands.
- ❑ All DOS disks have a root directory.
- ❑ The introduction of subdirectories and the release of DOS V3.0 introduced the path parameter in many commands.
- ❑ The MKDIR (MD) command makes new subdirectories.
- ❑ The CHDIR (CD) command changes default (working) directories.
- ❑ Use the PATH command to have DOS search for program and batch files.
- ❑ Use the APPEND command to have DOS search for data files.
- ❑ The XCOPY command can make new subdirectories while copying.
- ❑ The TREE command reports your directory tree structure.
- ❑ The RMDIR (RD) command removes empty directories.
- ❑ A DOS-specific subdirectory gives better control of external DOS command-path searches.
- ❑ Directories for applications software packages should be uniform in structure.
- ❑ Using a \TEMP directory can make file copying on a single floppy drive system easier.

- Using a \KEEP directory as part of a file's housekeeping method can keep directories less cluttered.

You are more likely to use the directory-level commands presented in this chapter than you are to use the disk-level commands presented in Chapter 5. The directory-level commands lend themselves to practice and experimentation. You can use your existing directory layout to try variations of these commands. The concept of the path will be important to your understanding the file-level commands presented in the next chapter. You may want to refer back to this chapter from time to time as you work with the commands that take directory parameters.

7

Managing Your Files

Chapter 5 examines the commands you use to manage your PC at the disk level. Chapter 6 presents a discussion of the DOS hierarchical file system as well as the commands that manage the file system's directories. This chapter advances the disk and directory theme of the last two chapters to the theme of managing files. In all likelihood, you spend more time in DOS activities that manage files than you do in all other DOS activities combined.

File-management commands are fundamental to even the most primitive use of a PC. Most elementary level texts tutor DOS beginners in the use of file-level commands in early chapters. Chances are good that you learned the basics of copying, deleting, and renaming files before you learned about directories and partition tables. Now that you have some DOS experience, you can take the time to review the fundamentals of the file-management commands and expand your understanding of files and their management.

This chapter leads off with a review of files from DOS's point of view as well as from the user's point of view. The chapter then focuses on the commands used in file management. The commands covered in this chapter include the following:

- ATTRIB
- RENAME (REN)
- ERASE (DEL)
- TYPE
- COMP and FC
- COPY
- VERIFY
- BACKUP
- RESTORE

Reviewing the Concept of Files

A file is the smallest unit of disk-based information that a computer user can manage. Every file has a name; DOS uses the file name to identify the information that the file contains. As a user, you view a file as a single named group of information. As a disk operating system, DOS views a file as a collection of allocated clusters which are accessible through a directory entry and the file allocation table. As a user, you concentrate on creating and using files as an extension of your applications programs. As a disk operating system, DOS concentrates on providing a uniform set of file-level services to applications programs and to DOS commands.

Reminder:
The DOS operating system occupies the middle of the layered computer, between the innermost layer of physical hardware and the outermost layer of applications programs.

The innermost layer of the computer is comprised of physical hardware; the outermost layer is comprised of applications programs. DOS, with its file services and commands, falls in the middle of this layered computer. When you use DOS commands to manage files, you do not need to know how the inner layers of DOS support those commands. Yet, the details of these inner workings manage to come to the surface in the form of rules and syntax. The rules governing DOS file names, for example, are mandated by the way DOS lays out its directories and by the way COMMAND.COM reads the commands from the keyboard. Rules are easier to understand and remember when you see their underlying logic. The following sections explain file rules as well as the underlying logic of those rules.

Selecting File Names

Reminder:
Every file created by DOS is given a name that is placed in one of the directories.

DOS ensures that every one of your files has a name. In fact, DOS provides no method of putting file data on a disk without a file name. When a file is created, either by your applications software or a DOS file-service routine, DOS places the name of a file in one of its directories.

DOS provides room in a directory entry for eight characters of a file-name prefix and three characters of a file-name suffix. The file-name prefix is what you think of as the *root* file name; the suffix is the file-name extension or simply *extension*. When you include a file name in a command, you place a period between the file name and extension. In the directory, DOS does not store the period character. DOS separates the file name from the extension internally by knowing the position of the extension in the directory field.

DOS could have been designed so that a file name consisted of seven characters of file name and four characters of extension—or nine characters of file name and two characters of extension. The UNIX operating system, for example, has no provision for file-name extensions. The OS/2 operating system allows file names that can include dozens of characters. DOS, however, allows up to eight characters of file name and up to three characters of extension because a DOS

directory can accommodate those numbers. DOS utilizes the file-naming convention of the once dominant CP/M operating system. DOS designers felt that using the CP/M file-name convention would allow an easier transition from CP/M computers to DOS computers.

Recall that characters are the ASCII-code representations of a byte of data. The name field of a directory entry is 11 bytes long (8 for file name plus 3 for extension). DOS accepts for file names most characters you would use for "everyday" names. You can use in file names the upper- and lowercase letters a through z; DOS automatically stores them in uppercase format in a directory entry. The numeric characters 0-9 can be used as well as the alternate graphics characters of a PC's keyboard. (The alternate characters, ASCII values 128-255, are produced by holding down the Alt key while pressing the number keys on the numeric keypad for the desired ASCII value.) Many punctuation characters are also allowed in file names.

Reminder: DOS stores lowercase alphabetic letters in uppercase format in the directory entry.

Following are the rules for file names.

Rules for File Names

1. A file name must have the following:
   a. A root name of one to eight characters
   b. An optional extension of one to three characters
   c. A period between the root name and the extension name if an extension is used
2. The following characters are *allowed* in a file name:
   a. The letters A to Z. (Lowercase letters are transformed automatically into uppercase.)
   b. The numbers 0 to 9.
   c. The following special characters and punctuation symbols:

      $ # & @ ! & ( ) - { } ' _ ~ ^ `
3. The following characters are *not allowed* in a file name:
   a. Any control character, including Escape (27d or 1Bh) and Delete (127d or 7Fh)
   b. The space character
   c. The following special characters and punctuation symbols:

      + = / [ ] " : ; , ? * \ < > |
4. If DOS finds an illegal character in a file name, it usually stops at the character preceding the illegal one and uses the legal part of the name. (Some exceptions are noted in this chapter.)

5. A device name can be part of a root name but cannot be the entire root name. For example, CONT.DAT or AUXI.TXT are acceptable; CON.DAT or AUX.TXT are not.
6. Each file name in a directory must be unique.
7. A drive name and a path name usually precede a file name. (Path names are discussed in Chapter 6.)

To understand why DOS objects to some characters in file names, you must look below the surface of DOS. Certain characters are not allowed because DOS will not pass those characters from the keyboard to the command or external command program that controls a file's name. You cannot, for example, use the Ctrl-G (^G) character in a file name. Ctrl-G tells an input device to sound the computer's bell or beep signal. The Escape character produced by the Esc key is another character that DOS will not accept as part of a file name. DOS interprets the Esc key as your request to cancel the command line and start again.

Reminder: *DOS does not allow control characters in a file name.*

Another example of a disallowed character is Ctrl-S (^S). DOS and other operating systems use Ctrl-S to stop the flow of characters from the input device to the output device. Ctrl-S stops a scrolling screen, for example. If you press Ctrl-S while entering a file name, DOS assumes that you are entering an input-stopping character, not part of a file name. When you type **DIR** immediately followed by Ctrl-S, and then press Enter, DOS does nothing. DOS sees the ^S and Enter sequence as a stop, and then start, flow. DOS thinks that the Enter key press is the signal to start sending keystrokes again. The temporarily stopped command processor then executes the DIR command.

DOS also does not allow in a file name any characters that COMMAND.COM and external command programs use to divide the parameters and switches from the command in a command line. Because DOS must break out, or *parse*, the components of the command line, certain predefined characters are seen by DOS as delimiters of parameters or parts of a parameter. The backslash character (\), for example, separates directory names in a path specifier. DOS always reads the backslash on the command line to indicate part of a path.

Reminder: *DOS does not disallow spaces embedded in file names that are created by applications programs.*

Surprisingly, DOS does not disallow spaces embedded in file names that are created by applications programs. Most programs ensure that file names are legal before asking DOS to create a file using the name. Others do not. Microsoft QuickBASIC, a program used for writing BASIC programs, is an example. A program written in QuickBASIC allows files to be created with spaces embedded in a file name. The following directory listing of files shows two entries that include spaces and one without spaces.

```
 Volume in drive C is LAP TOP
 Directory of  C:\QUICK

1234567  123         13   8-25-89   7:58a
1 2 3 4  123         14   8-25-89   7:58a
S P A CE 123         14   8-25-89   7:59a
    3 File(s)   1996800 bytes free
```

DOS accepts the file names that contain embedded spaces and displays them in a directory listing with no objections.

Reminder:
It is impractical to use a space in a file name even though some programs allow the use of spaces in directory listings.

If, however, you try to use one of the files with embedded spaces in a command line, DOS objects with the following message:

```
Invalid number of parameters
```

You cannot use COMMAND.COM to get to one of the files with embedded spaces in its name, so it is impractical to use a space, even when a program allows it.

Avoiding Reserved Names

DOS reserves names for its built-in character devices. You recall from Chapter 4 that DOS can treat some devices in a PC in a high-level way by accepting their names as input or output parameters in a command line. Before DOS looks for a file using the file name parameters from a command line, DOS checks whether the name is a device name. Table 7.1 lists the DOS character device names and their purposes.

Caution:
Avoid using file names reserved for DOS devices.

Table 7.1
DOS Device Names

Device Name	*Purpose*
COM*x* or AUX	Identifies the serial communication port(s)
LPT*x* or PRN	Identifies the parallel printer port(s)
CON	Identifies the screen and keyboard
NUL	Identifies the "throw away" or "do nothing" device

If you intend for DOS to use one of these device names as a file name, you may be disappointed with the outcome of your command. DOS will intercept the device name, even if you add an extension, and try to use the device—not the file that you intend—to complete the command. The rule on the reserved device names of DOS is simple: Use a device name only as a device parameter in a command. Never expect to write a disk file with a root file name that is a device name.

Reminder:
Use only a device name as a device parameter in a command.

Observing File-Naming Conventions

A convention is an informal rule that is not explicitly enforced. DOS file names follow certain conventions. Although you can use any file name that passes character rules and device-name rules, you will want to observe file-name conventions whenever possible. You can, for example, name a memo file using a .BAT extension; DOS would not know that you were not saving a conventional batch file. As long as you do not try to execute the memo as a batch file, DOS is happy. If you try to execute the memo file, however, DOS sees the .BAT extension and tries to execute it. Of course, the memo cannot be executed.

You can also name an .EXE file with a .COM extension. Although both files are executable, they have internal differences. DOS does not take the extension's name to mean that the file is indeed an .EXE or .COM file. DOS inspects a key part of the file before deciding how to load and execute the program file. If you name a spreadsheet file as an .EXE or .COM file, for example, DOS is not fooled into trying to execute a nonprogram file.

Reminder: *Avoid giving your file names extensions that are used conventionally by DOS programs and other applications programs.*

Many software manufacturers use certain extensions for special files in their applications programs. To avoid confusion as to the contents of a file, avoid using certain extensions. Table 7.2 lists conventional file-name extensions and their meanings.

Table 7.2
Common File-Name Extensions

Extension	*Common Use*
.ARC	Archive (compressed file)
.ASC	ASCII text file
.ASM	Assembler source file
.BAK	Backup file
.BAS	BASIC program file
.BAT	Batch file
.BIN	Binary program file
.C	C source file
.CBL	COBOL source file
.CFG	Program configuration information
.CNF	Program configuration information
.CHP	Chapter file (Ventura Publisher)
.COM	Command (program) file
.CPI	Code page information file (DOS)
.DAT	Data file
.DBF	Database file (dBASE III)
.DCT	Dictionary file
.DEV	Program device driver file
.DIF	Data Interchange Format file

Extension	*Common Use*
.DOC	Document (text) file
.DRV	Program device driver file
.DTA	Data file
.EXE	Executable program file
.FNT	Font file
.IDX	Index file (Q&A)
.IMG	GEM image (graphics) file
.HLP	Help file
.KEY	Keyboard macro file (ProKey)
.LET	Letter
.LST	Listing of a program (in a file)
.LIB	Program library file
.MAC	Keyboard macro file (Superkey)
.MAP	Linker map file
.MEU	Menu file (DOS Shell and SELECT)
.MOS	Mouse drive file (DOS Shell)
.MSG	Program message file
.NDX	Index file (dBASE III)
.OBJ	Intermediate object code (program) file
.OLD	Alternative extension for a backup file
.OVL	Program overlay file
.OVR	Program overlay file
.PAK	Packed (archive) file
.PAS	Pascal source file
.PCX	Picture file for PC Paintbrush
.PIF	Program Information File (TopView/Windows)
.PRO	Profile configuration file (DOS GRAPHICS)
.PRN	Program listing for printing
.PS	PostScript program file
.RFT	Revisable Form Text (Document Content Architecture)
.SAV	Alternative extension for a backup file
.SYS	System or device driver file
.STY	Style sheet (Ventura Publisher)
.TIF	Picture file in Tag Image Format
.TMP	Temporary file
.TST	Test file
.TXT	Text file
.$xx	Temporary or incorrectly stored file
.WK1	Worksheet file (1-2-3, Release 2)
.WK3	Worksheet file (1-2-3, Release 3)
.WKQ	Quattro spreadsheet file
.WKS	Alternative extension for a worksheet file

Avoiding Bad File Names

When you enter file names in the command, DOS may not simply reject the bad file name. In some cases, DOS takes some unusual actions. To create a new file from the file TEST.TXT, you can, for example, enter the following command:

COPY TEST.TXT 123456789.1234

Reminder:
DOS truncates to 8 characters for the root and 3 characters for the extension any file name that is entered with more than the legal number of characters.

Notice that both the root file name and the extension have an extra character. You might predict that DOS will issue some message to warn that the file name and extension are both too long. DOS does not issue a warning but silently truncates the file name to eight characters and the extension to three characters. The resulting file is named `12345678.123`. DOS always discards excess characters in the file name and extension and uses the remaining characters as the file parameter.

Illegal characters in a file name can also stop DOS from carrying out a command. The following interaction with DOS is an example.

COPY 12345678.123 1[3.123

```
File creation error
        0 File(s) copied
```

As you might expect, the illegal character [in the target file's extension stopped DOS from completing the copy operation. If you assume that the [character will always produce a file-creation error when the character is used in a file name, however, you may be surprised. The following interaction with DOS shows a surprising result.

COPY 12345678.123 87654321.12[

```
File cannot be copied onto itself
          0 file(s) copied
```

DOS sees the target file name as being the same as the source file name. The only peculiar aspect of either file name is the [character in the last character position of the target file's extension. Although the error message this command invokes is not what you would predict, DOS does stop the command.

If you use the same [character in a reporting command such as DIR, you might predict that DOS would issue an error message because of the illegal character in the file parameter. The following sequence shows that no error message is issued.

DIR [

```
Volume in drive C is LAP TOP
Directory of  C:\DATA

File not found
```

DOS performed a directory search for something and found nothing. No error message was issued even though the only file parameter was an illegal character.

Not every character that is illegal in a file name stops DOS from carrying out a command. In the following example, the semicolon character is at the end of the destination file's extension.

```
COPY 12345678.123 12345678.12;
        1 File(s) copied
```

DOS carries out the COPY command and creates a new file from the destination parameter. To see the new file's name, issue the DIR command and view the listing:

```
 Volume in drive C is LAP TOP
 Directory of  C:\DATA

TEST     TXT         5   8-25-89    9:25a
12345678 123         5   8-25-89    9:32a
12345678 12          5   8-25-89    9:35a
        3 File(s)  1978368 bytes free
```

DOS added the `12345678.12` file. DOS reads the ; in the command as a closing delimiter. The remaining part of the target file parameter became the new file's name.

Not many DOS users with some experience would use illegal characters in a file name on purpose. Typos, however, can easily creep into your command lines and introduce illegal characters into file-name parameters. Just remember that the reaction of DOS to an illegal file name isn't always predictable.

Finding the End of a File

Knowing how DOS finds the end of a file may seem like an unnecessary technical detail, but this knowledge can help you understand the concept of copying from devices and copying from files. As discussed in Chapter 2, DOS does not impose any form on the contents of a file. To DOS, a file is simply a stream of bytes. Remember that a byte can represent more than one kind of value. A byte can be encoded to represent a number, a computer instruction, a table, or even an ASCII character. Usually, computer professionals divide the representation categories of a byte into two general types: ASCII and binary.

Reminder:
A byte can be classified into two general types: ASCII and binary.

An ASCII byte contains in its 8 bits a bit code that represents one of 256 possible ASCII characters. Files that are composed of bytes, all of which are ASCII representations, are called ASCII text files. Not all ASCII codes, however, represent readable text characters such as letters, numbers, and punctuation symbols.

Some ASCII characters represent device-control codes, which a device detects and uses to control its operation. Ctrl-S is such a control code; a device that detects Ctrl-S in a character stream stops sending its stream of characters. You use Ctrl-S to stop a display from scrolling off the screen. When the control characters are not being used for controlling a device, they can represent special characters on-screen, such as a smiling face or musical notes. Other ASCII representations make special characters on your screen, but have no device-control representations.

Reminder:
ASCII text is not always readable when displayed by the TYPE command.

ASCII text is not always readable on your PC display. Figure 7.1 shows the contents of a paragraph (which was prepared in WordStar) as displayed by the TYPE command. The figure shows the alternate characters at the end of each word. To DOS, the file is a stream of bytes. The TYPE command makes the assumption that the bytes are ASCII representations.

Fig. 7.1

The ASCII display of a WordStar file.

```
Programs anΣ externa= commands ca€ instruc∫ DO" t∩ internall· ì
                                                              wor∂ wit¤ inpu∫ a
nΣ outpu∫ bytes i€ a€ ASCI‖ mode« I€ a€ ASCI‖ ì
                                                        mode¼ DO" recognizes on◦ ASCI‖ co
ntro= characte² as meanin┬ "th◦ ì
                                      enΣ o╡ file"« Ctr=┌ is DOS's enΣ o╡ fil◦ marke
r« Applicatio€ ì
                  programs tha∫ creat◦ ASCI‖ tex∫ files senΣ ╨ strea≠ o╡ character
s ì
   t∩ DO" fo² DOS's fil◦ service« Th◦ applications plac◦ ╨ Ctr=┌ i€ ì
                                                                    thei² char
acte² outpu∫ strea≠ t∩ indicat◦ t∩ DO" tha∫ th◦ enΣ o╡ ì
                                                          th◦ characters tha∫ wil=
 mak◦ u≡ th◦ fil◦ has bee€ reached« DO" ì
                                                  use¼ th◦ Ctr=┌ characte² as ╨ signa= t
∩ clos◦ th◦ fil◦ anΣ ì
                              updat◦ th◦ file's director· entry«
```

When the most significant bit (the one with the highest value) in a ASCII byte is set (has a value of 1), the 128 distinct combinations of the lower 7 bits can represent special graphic-like characters. Some word processors may set the 8th bit of a byte to transform a printable character into a special character. The word processor does not print the special character as a special character, but rather uses the 8th bit internally to indicate that the special character is the last character of a word or paragraph. When the character is printed, the 8th bit is ignored, and the printable character represented by the remaining 7 bits is printed.

Programs and commands like COPY can instruct DOS to internally work with input and output bytes in an ASCII mode. In ASCII mode, DOS recognizes one ASCII control character—the ^Z—as meaning "the end of the file." Applications programs that create ASCII text files send a stream of characters to DOS for DOS's file service. The applications place a Ctrl-Z in their character-output stream to indicate that the end of the characters that will make up the file has been reached. In ASCII operations, DOS uses the Ctrl-Z character as a signal to close the file and update the file's directory entry.

Cue:
In ASCII mode, the ^Z character represents the end of the file.

If an application uses these file services in a non-ASCII or binary mode, the application must explicitly tell DOS to close the new file even if the application sends DOS a Ctrl-Z. The application can send additional bytes of file data to DOS following a Ctrl-Z character. The application is not obligated to place the Ctrl-Z at the end of the physical file (the point in the data stream where DOS is told to close the file). In fact, between the Ctrl-Z and the end of the physical file, the application can pad several hundred bytes with filler values. In the binary mode, DOS does not care that it has encountered the Ctrl-Z character.

Cue:
Most DOS file commands deal with files in a binary mode instead of an ASCII mode to avoid a case of mistaken identity.

Most DOS file commands, by default, deal with files in binary instead of ASCII mode to avoid a case of mistaken identity. Remember that numeric values, instructions, and other data are alternative representations of the value of a byte. There is a good chance that a byte from a numeric value or instruction has the same value of the Ctrl-Z in its ASCII representation. A binary file might contain several bytes that could represent an ASCII Ctrl-Z, but are actually data or instructions in binary representation. In ASCII mode, DOS would mistake these values as the end-of-file marker and prematurely close the file. The result of this mistaken identity would be the loss of file data.

Understanding Devices and the End-of-File Marker

Recall from Chapter 4 that DOS's built-in character devices can be used as source and destination parameters in command lines. When an input device such as CON is used for direct or redirected input in a command, DOS needs some way to know when the input character stream is finished and the file should be closed. The device drivers for the built-in devices of DOS work in ASCII mode; they use the Ctrl-Z character as the end-of-file indicator when the devices are used as parameters in DOS commands.

The use of the COPY command with the CON keyboard device is a good example of the need for an end-of-file marker. Although the COPY command is covered in detail later in this chapter, a brief example here will help you see the role of ^Z. When you issue the command **COPY CON MYFILE.TXT**, DOS uses the console device, CON, as the source file and expects to copy ASCII charac-

ters from keystrokes entered in the target file. As you press the characters that you want included in the new MYTEXT.TXT file, DOS buffers the characters internally.

At some point, you finish providing characters for the new file and want to issue more DOS commands. Unless you signal DOS that you are finished with the COPY command, DOS will continue filling the internal buffer with every letter that you type. Without pressing a Ctrl-Z (or F6, which sends Ctrl-Z and then Enter), you would be hung in the COPY command. When you press Ctrl-Z and then Enter, DOS creates the new file's directory entry and copies your keystrokes out of its internal buffer to the file. DOS's last step is to close the file and update the directory for the time, date, and file size entries.

Keep in mind the topics introduced in these first few sections when you are working with files in DOS. Knowing what DOS expects in a file name is important. Knowing how some DOS commands work with Ctrl-Z and ASCII is also necessary for understanding the more advanced commands.

Manipulating File Attributes with ATTRIB

Cue:
The external command ATTRIB enables you to manipulate the file's read/write and archive attributes.

Remember that each directory entry provides some fields to record a file's attributes. The external command ATTRIB enables you to manipulate the file's read/write and archive attributes. Both of these attributes are given default values by DOS when the file is added to the directory.

Issuing the ATTRIB Command

You can issue the ATTRIB command in three basic ways.

1. Use the following syntax form to receive a screen report of a file's current attribute values:

   ATTRIB *d:path***filename.ext**/S*

2. Use the following syntax form to set (turn on) a file's attribute(s):

   ATTRIB *+R +A d:path***filename.ext** */S*

3. Use the following syntax form to reset (turn off) a file's attribute(s):

   ATTRIB *−R −A d:path***filename.ext** */S*

d: is the optional drive that holds the disk containing the selected files, and *path*\ is the optional directory path that contains the selected files.

filename.ext, the selected file specifier, can contain wild cards. A file name must be specified.

The optional */S* switch (V3.3 and later versions only) instructs ATTRIB to additionally process files that match the file specifier in all *subdirectories* of the path directory.

The *+R* and *+A* attribute indicators instruct ATTRIB to set (add) the read-only and archive attributes respectively.

The *−R* and *−A* attribute indicators instruct ATTRIB to reset (clear) the read-only and archive attributes respectively.

The R and A attributes can be specified individually or together in the command line.

The R and A attributes can have the same action (+R and +A or −R and −A) or opposite actions (−R and +A or +R and −A).

If neither an R nor an A attribute specifier is included in the command line, the ATTRIB command produces a screen report showing the attribute status of desired files.

Understanding the Operation of ATTRIB

DOS provides an attribute in the directory entry of each file to indicate the archive status of the file. By DOS's action, each file has its archive attribute turned on when the file is stored in the directory. Any time that DOS writes data to a file or shortens (truncates) a file, DOS turns on the archive attribute. DOS V3.2 and later versions allow the ATTRIB command to manipulate the archive attribute. If your version of ATTRIB manipulates archive attributes, you can also use the command on files created with earlier versions of DOS.

Establishing Read-Only Files

Cue:
The read/write attribute controls the capability of DOS to overwrite or erase a file.

The read/write attribute controls the capability of DOS to overwrite or erase a file. When DOS (or an application through DOS) adds a file to a directory, the read/write attributes are established so that the file can be overwritten or erased. The default value of the read/write attribute allows DOS commands to perform destructive operations on the file. If you change the read/write attribute of a file to make the file "read-only," the DOS commands that normally overwrite or erase files will not affect the read-only file.

After you use the ATTRIB command to mark a file as read-only, you can use the ATTRIB command again to mark the file as read/write. The file will then be subject to DOS commands that can overwrite or erase the file.

Marking a file as read-only protects the file in a similar way to the manner in which write-protecting a disk protects the contents of a disk. You can use the ATTRIB command effectively to write-protect important files and have insurance that an errant COPY or ERASE command will not destroy the marked files.

Displaying Archive Attributes

Cue:
ATTRIB reports the current attributes of a file when you issue the command without an attribute specifier.

ATTRIB is one of the DOS commands that can report information. ATTRIB reports the current attributes of a file when you issue the command without an attribute specifier. The resulting file list will show the read/write and archive attribute settings of specified files. When you issue ATTRIB in a report syntax, all attributes of reported files remain unchanged.

The attribute-report syntax of ATTRIB is useful for determining which files will be processed or have had their attributes modified by a BACKUP, RESTORE, XCOPY, or ATTRIB command. You should view a reported list of file attributes before you issue a command that will change those attributes.

Understanding the General Rules of ATTRIB

- The A attribute is available with DOS V3.2 and later versions.
- The /S switch is available with DOS V3.3 and later versions.
- You must include the A or R attribute specifier to modify a file's existing attributes with the ATTRIB command.
- The ATTRIB command does not affect a file's data. It affects only a file's specified attributes.
- You cannot modify a file's hidden or system attributes with ATTRIB.
- You must provide a file specifier, but drive and path specifiers can be omitted. The current values of drive and path will be used by default if either specifier is omitted from the command.
- The /S switch allows ATTRIB to process all subordinate directories in addition to the specified (or default) directory.
- You can mix the set (+) and reset (−) attribute indicators in the same command.

Cue:
The archive attributes work with the XCOPY, BACKUP, and RESTORE commands to determine whether a file can be processed.

Using Archive Attributes with XCOPY, BACKUP, and RESTORE

An important benefit of archive attributes is their capability to tell XCOPY, BACKUP, and RESTORE that a file should be processed. The XCOPY command

covered in Chapter 6, and the BACKUP and RESTORE commands covered later in this chapter, can optionally use the archive attribute of a file to select files for inclusion in their operation. Both XCOPY and BACKUP can clear the archive attribute of a file when the commands process a file. (For details of how each command works, see the appropriate chapter or the Command Reference.)

XCOPY and BACKUP both accept the /M switch, which means "operate on *modified* files." In other words, either command issued with the /M switch limits the source files to those which have the archive attributes set (turned on). Remember that DOS turns on the archive attribute of a file when the file is created or modified.

Reminder:
You can use XCOPY and BACKUP with the /M switch to copy or back up changed files.

You can use XCOPY and BACKUP with the /M switch to copy or back up changed files. You do not have to check directory dates or remember your work activities to determine which files you should place in another directory or disk for safekeeping. The archive attribute and the /M switch make the decision for you. Once the files are processed using the /M switch, the archive attributes of the source files are reset (turned off).

You might use XCOPY several times between backups. When **XCOPY /M** resets (turns off) the archive attributes of the files copied, the **BACKUP /M** command cannot find those files. Your next use of BACKUP might miss files that changed since the last backup. To avoid this possibly confusing situation, you can use **XCOPY /A** which also copies files whose archive attribute is set. **XCOPY /A**, however, does not reset the archive attributes of the files copied. A subsequent **BACKUP /M** command will find and include the files even though **XCOPY /A** processed the same files first.

RESTORE uses the optional /M switch to selectively restore files from a backup set by examining the archive attribute of the target file. If the archive attribute is set, RESTORE with /M will restore the file from the backup. If the archive attribute is not set, **RESTORE /M** will not restore the file from the backup. The RESTORE command is discussed in detail later in this chapter.

You can directly manipulate the archive attribute of a file with the ATTRIB command. In modifying a file's archive attribute, you can override the default setting of the file's archive attribute. Even if XCOPY or BACKUP have cleared a file's archive attribute, ATTRIB can set the attribute manually.

In fact, a useful way to approach the BACKUP and XCOPY commands is to select files that you want to include in their operation by changing the file's attributes. To include all MEM files in the /MEMO subdirectory in an **XCOPY /M** operation, you first turn on all archive attributes for the MEM files with the following command:

```
ATTRIB +A \MEMO\*.MEM
```

XCOPY /M will include all of the MEM files.

Using ATTRIB To Write-Protect Applications Program Files

Cue:
You can use the ATTRIB command to "write-protect" certain files by using one command.

If you have created a \DOS directory for your DOS external program files, you can use the ATTRIB command to "write-protect" the external command files using one command. Because you are unlikely to want to erase an external command from the \DOS directory, you can issue the following command:

ATTRIB +R C:\DOS*.*

This command assumes that the default search path can locate the external ATTRIB command. Because the disk and path specifiers are included in the command, you can issue this command from any disk or directory.

The *.* wild-card file specifier tells ATTRIB to process all files in C:\DOS. The +R attribute tells ATTRIB to mark all processed files as read-only. All files in C:\DOS will be bypassed in any ERASE or COPY command that specifies the files. If you attempt to erase any file from the \DOS directory, the following message will appear:

```
Access denied
```

To allow a COPY or ERASE command to modify or erase a read-only file, you can issue the ATTRIB command using the file's name and the −R attribute specifier. To make the RECOVER.EXE command erasable after the previous command, you can log to C:\DOS and issue the following command:

ATTRIB −R RECOVER.EXE

You can now erase or overwrite the RECOVER.EXE file in C:\DOS.

Cue:
Use the ATTRIB command to write-protect the core files of your applications programs.

Many applications programs such as dBASE IV copy their main program and support files to a specific directory during the installation process. These core package files are seldom modified after installation. You will find an applications program's core files to be ideal candidates for read-only protection.

To set the read-only attributes of all dBASE IV core package files after you install the package, first log to the dBASE IV directory or the directory in which your files are stored. Then issue the following command:

ATTRIB +R *.*

The same basic command can change any directory's files to read-only. Be sure that you are in your intended directory.

When the command is executed, the files in the \DBASE directory are protected. You can verify that the read-only attribute is set by issuing the ATTRIB command for an extension that will show a sampling of the files. For dBASE, the

.OVL files are a good choice. From any directory on drive C:, you can issue the following command:

```
ATTRIB \DBASE\*.OVL
```

You will then see the following screen report:

```
A     R  C:\DBASE\DBSETUP.OVL
A     R  C:\DBASE\DBASE3.OVL
A     R  C:\DBASE\DBASE1.OVL
A     R  C:\DBASE\DBASE2.OVL
A     R  C:\DBASE\DBASE5.OVL
A     R  C:\DBASE\DBASE4.OVL
A     R  C:\DBASE\PROTECT.OVL
A     R  C:\DBASE\RUNTIME1.OVL
A     R  C:\DBASE\RUNTIME3.OVL
A     R  C:\DBASE\RUNTIME2.OVL
A     R  C:\DBASE\RPROTECT.OVL
A     R  C:\DBASE\RUNTIME4.OVL
```

Notice that each file is listed with its full path name and that both the archive and read-only indicators are shown for all files. If either reported attribute is not set (is cleared), that attribute will not be indicated beside the file name.

Using ATTRIB with the XCOPY Command

The archive attributes of the files in a directory can be useful when you use XCOPY to maintain a current set of files on a diskette. In any directory in which data files or program source files are being changed frequently, you can use XCOPY to copy only the changed files to a diskette. Of course, you must have enough capacity on the diskette to hold all the files XCOPY will copy.

This example illustrates the use of the archive attribute in XCOPY. Suppose that you have a hard disk with a PROLOG programming language package installed. A directory named \PROLOG\PROGRAMS holds the .PRO program source text files as well as some miscellaneous files. Suppose that you want to keep a current version of all of the .PRO source files on a diskette. You copy .PRO files that have changed in the course of a day to the diskette before you leave work. You do not alter all the source files each day, so you want to copy only the files that have changed during the day. When you first decide to make a daily copy, you can look at the archive attributes of all the \PROLOG\PROGRAMS files by using the following command:

```
ATTRIB \PROLOG\PROGRAMS\*.*
```

DOS reports the files as follows:

```
A          C:\PROLOG\PROGRAMS\BROWSER.PRO
A          C:\PROLOG\PROGRAMS\DBATEST.PRO
A          C:\PROLOG\PROGRAMS\DIFF.HLP
A          C:\PROLOG\PROGRAMS\DIFF.PRO
A          C:\PROLOG\PROGRAMS\DUMPDBA.PRO
A          C:\PROLOG\PROGRAMS\D_CURVE.PRO
A          C:\PROLOG\PROGRAMS\FWGC.PRO
A          C:\PROLOG\PROGRAMS\GENI.GNI
A          C:\PROLOG\PROGRAMS\GENI.HLP
A          C:\PROLOG\PROGRAMS\GENI.PRO
A          C:\PROLOG\PROGRAMS\GEOBASE.DBA
A          C:\PROLOG\PROGRAMS\GEOBASE.HLP
A          C:\PROLOG\PROGRAMS\GEOBASE.INC
A          C:\PROLOG\PROGRAMS\GEOBASE.LAN
A          C:\PROLOG\PROGRAMS\GEOBASE.PRO
A          C:\PROLOG\PROGRAMS\GLOBALS.PRO
A          C:\PROLOG\PROGRAMS\GRAPDECL.PRO
A          C:\PROLOG\PROGRAMS\GRDEMO.PRO
A          C:\PROLOG\PROGRAMS\HANOI.PRO
A          C:\PROLOG\PROGRAMS\LINEINP.PRO
A          C:\PROLOG\PROGRAMS\LISTDBA.PRO
A          C:\PROLOG\PROGRAMS\MAIN.PRO
A          C:\PROLOG\PROGRAMS\MENU2.PRO
A          C:\PROLOG\PROGRAMS\MIXTURE.PRO
A          C:\PROLOG\PROGRAMS\MONSTERS.PRO
A          C:\PROLOG\PROGRAMS\NDXTEST.PRO
A          C:\PROLOG\PROGRAMS\PRNHND.PRO
A          C:\PROLOG\PROGRAMS\RANDOM.PRO
A          C:\PROLOG\PROGRAMS\REFSHND.PRO
A          C:\PROLOG\PROGRAMS\REGISTER.BIN
A          C:\PROLOG\PROGRAMS\REGISTER.HLP
A          C:\PROLOG\PROGRAMS\REGISTER.LOG
A          C:\PROLOG\PROGRAMS\REGISTER.PRJ
A          C:\PROLOG\PROGRAMS\REGISTER.SYM
A          C:\PROLOG\PROGRAMS\REGISTER.ZUP
A          C:\PROLOG\PROGRAMS\SCRHND.OBJ
A          C:\PROLOG\PROGRAMS\SEN_AN.HLP
A          C:\PROLOG\PROGRAMS\SEN_AN.PRO
A          C:\PROLOG\PROGRAMS\SEN_AN.SAN
A          C:\PROLOG\PROGRAMS\TDOMS.PRO
A          C:\PROLOG\PROGRAMS\TPREDS.PRO
A          C:\PROLOG\PROGRAMS\TRAPTEST.PRO
A          C:\PROLOG\PROGRAMS\VACATION.GNI
```

As you can see, all files in the directory have their archive attribute set. To copy only the .PRO files with their archive attributes set (in this instance, all .PRO files) to the diskette in drive A:, you issue the following command:

XCOPY \PROLOG\PROGRAMS*.PRO /M

The XCOPY command copies the .PRO files to the diskette in drive A: and resets the archive attribute of each file as the file is copied. When XCOPY completes the copy process, you can verify that the source files with .PRO extensions in \PROLOG\PROGRAMS have their archive attributes reset and that all other files' archive attributes are unchanged. You then issue the following command:

ATTRIB \PROLOG\PROGRAMS*.*

DOS will report the following:

```
        C:\PROLOG\PROGRAMS\BROWSER.PRO
        C:\PROLOG\PROGRAMS\DBATEST.PRO
A       C:\PROLOG\PROGRAMS\DIFF.HLP
        C:\PROLOG\PROGRAMS\DIFF.PRO
        C:\PROLOG\PROGRAMS\DUMPDBA.PRO
        C:\PROLOG\PROGRAMS\D_CURVE.PRO
        C:\PROLOG\PROGRAMS\FWGC.PRO
A       C:\PROLOG\PROGRAMS\GENI.GNI
A       C:\PROLOG\PROGRAMS\GENI.HLP
        C:\PROLOG\PROGRAMS\GENI.PRO
A       C:\PROLOG\PROGRAMS\GEOBASE.DBA
A       C:\PROLOG\PROGRAMS\GEOBASE.HLP
A       C:\PROLOG\PROGRAMS\GEOBASE.INC
A       C:\PROLOG\PROGRAMS\GEOBASE.LAN
        C:\PROLOG\PROGRAMS\GEOBASE.PRO
        C:\PROLOG\PROGRAMS\GLOBALS.PRO
        C:\PROLOG\PROGRAMS\GRAPDECL.PRO
        C:\PROLOG\PROGRAMS\GRDEMO.PRO
        C:\PROLOG\PROGRAMS\HANOI.PRO
        C:\PROLOG\PROGRAMS\LINEINP.PRO
        C:\PROLOG\PROGRAMS\LISTDBA.PRO
        C:\PROLOG\PROGRAMS\MAIN.PRO
        C:\PROLOG\PROGRAMS\MENU2.PRO
        C:\PROLOG\PROGRAMS\MIXTURE.PRO
        C:\PROLOG\PROGRAMS\MONSTERS.PRO
        C:\PROLOG\PROGRAMS\NDXTEST.PRO
        C:\PROLOG\PROGRAMS\PRNHND.PRO
        C:\PROLOG\PROGRAMS\RANDOM.PRO
        C:\PROLOG\PROGRAMS\REFSHND.PRO
A       C:\PROLOG\PROGRAMS\REGISTER.BIN
A       C:\PROLOG\PROGRAMS\REGISTER.HLP
```

```
A          C:\PROLOG\PROGRAMS\REGISTER.LOG
A          C:\PROLOG\PROGRAMS\REGISTER.PRJ
A          C:\PROLOG\PROGRAMS\REGISTER.SYM
A          C:\PROLOG\PROGRAMS\REGISTER.ZUP
A          C:\PROLOG\PROGRAMS\SCRHND.OBJ
A          C:\PROLOG\PROGRAMS\SEN_AN.HLP
           C:\PROLOG\PROGRAMS\SEN_AN.PRO
A          C:\PROLOG\PROGRAMS\SEN_AN.SAN
           C:\PROLOG\PROGRAMS\TDOMS.PRO
           C:\PROLOG\PROGRAMS\TPREDS.PRO
           C:\PROLOG\PROGRAMS\TRAPTEST.PRO
A          C:\PROLOG\PROGRAMS\VACATION.GNI
```

Notice that only the .PRO extension files have their archive attributes reset. The other files' attributes remain unchanged because XCOPY did not process the non-.PRO files.

During the day, suppose that you use the EDLIN text editor to edit the BROWSER.PRO file. When DOS updates the file's contents, DOS updates the archive attribute in the file's directory entry by turning the attribute back on. You can confirm that the file's archive attribute was set by issuing the following command:

ATTRIB \PROLOG\PROGRAMS\BROWSER.PRO

DOS will report as follows:

```
A          C:\PROLOG\PROGRAMS\BROWSER.PRO
```

The archive attribute has been set because the file was modified by EDLIN. At the end of the day, you issue the daily copy command you have established as follows:

XCOPY \PROLOG\PROGRAMS*.PRO A:/M

As the XCOPY command is executed, you will see the following:

```
Reading source file(s). . .
BROWSER.PRO
        1 File(s) copied
```

Of all of the PRO extension files in the directory, only BROWSER.PRO was processed by the XCOPY command.

Making COMMAND.COM Read-Only with ATTRIB

COMMAND.COM is an important program file. To protect the file against accidental erasure or overwriting, you can make COMMAND.COM read-only.

Assume that COMMAND.COM is in your root directory, and then issue the following command:

ATTRIB +R \COMMAND.COM

COMMAND.COM is now a read-only file. To verify that the file has its read-only attribute set, issue the following command:

ATTRIB \COMMAND.COM

DOS replies as follows:

```
R   C:\COMMAND.COM
```

If you then issue the ERASE command with COMMAND.COM as the file specifier, DOS reports as follows:

```
Access Denied
```

DOS will not erase the file. The same type of command can make another file read-only. By using wild cards in the file specifier, you can make groups of files read-only. To return a file's read-only status back to read-write, you reverse the process. In the case of COMMAND.COM, you issue the following command:

ATTRIB −R \COMMAND.COM

The file can now be erased or overwritten.

Changing File Names with RENAME (REN)

Cue:
The RENAME command changes the name of an existing file.

File renaming in DOS is a straightforward process. If you aren't satisfied with a file's name, you simply change it. The RENAME command is your DOS tool to alter the name of an existing file. The two command names, RENAME and REN, work in identical ways.

RENAME is useful in situations other than the occasional renaming of a file when its current name isn't just right. When you use wild cards with RENAME, it can be a very powerful command.

Issuing the RENAME Command

The command name RENAME and REN are identical in operation; you can use either name in a command. The syntax for the RENAME command is as follows:

RENAME *d:path***filename1**.ext1* **filename2**.*ext2*

The file to be renamed is on the disk in the default drive or the optionally listed drive *d:*.

The file to be renamed is in the default directory or the directory listed in *path*\.

The file is currently named **filename1**.*ext1*. Wild cards are allowed in the full file-name specifier in the file name, the extension, or both.

filename2.*ext2* is the new name for the file. The new file name can be literal (an actual new name) or contain wild cards. You cannot specify a drive or a path for **filename**.*ext2*.

Understanding the Operation of RENAME

Reminder: *Even though RENAME changes the name in a file's directory entry, the file itself and its physical location on the disk remain unchanged.*

RENAME changes the name in a file's directory entry. The file itself and its physical location on the disk, however, remain unchanged. RENAME takes the new file name either literally from the command line or by filling in wild cards and scanning the directory that holds the file(s) to be renamed for duplicate file names. Because a directory cannot have a duplicate file name, RENAME will not change a file to the new file name if the new file name already exists.

If you use wild cards in the old file name and more than one existing file matches the pattern you specify, RENAME processes the old files in order. In other words, when DOS performs a top-down search of the directory, the files are renamed in the order in which they are encountered. Before each renaming of two or more files matching the wild-card pattern of the previous file specifier, DOS ensures that the file name it is processing will have a unique new name.

Understanding the General Rules of RENAME

- You can use the command names REN and RENAME interchangeably. Both command names produce identical results.
- You *must* supply an old file name and a new file name on the command line. Both the old file name and the new file name can contain wild cards for pattern matching.
- The renamed old file will remain on the same disk in the same path after renaming.
- You cannot use RENAME to move a file from a directory or a disk to another directory or disk. To move a file, copy it to the new destination and erase the original file from the source location.

Reminder: *You cannot use RENAME to move a file from a directory or a disk to another directory or disk.*

- RENAME does not issue a message if the file-renaming was successful. RENAME does issue messages in the following instances:

  1. You included an invalid parameter on the command line.
  2. The old file name does not exist in the optional specified full path or the default full path.
  3. The new file name duplicates an existing file name.

Renaming Files

The most controlled way to use the RENAME command is to issue the command with a literal old file name and new file name. You then do not have to be concerned about wild cards causing unwanted renaming of pattern-matching file names. Suppose that you have a sales report backup file, SALES.BAK, that has been prepared with your word processor. To give the file a more descriptive name, you can issue the following command:

```
RENAME SALES.BAK SALES_08.REP
```

You can be sure that only the desired file will be renamed because you literally specified the name of the file to be renamed. The resulting directory of the default directory will be as follows:

```
 Volume in drive C is LAP TOP
 Directory of C:\NAMES

.            <DIR>      8-29-89   9:29a
..           <DIR>      8-29-89   9:29a
SALES_08 REP      1664   8-01-89   2:51p
       3 File(s)   2009088 bytes free
```

After adding two more files to the default directory, you list the directory and see the following:

```
 Volume in drive C is LAP TOP
 Directory of  C:\NAMES

.            <DIR>      8-29-89   9:29a
..           <DIR>      8-29-89   9:29a
SALES_08 REP      1664   8-01-89   2:51p
SALES_09 REP       128   9-23-89   1:34p
EXPENSE  YTD     25372   9-23-89   8:00p
   5 File(s)   1978368 bytes free
```

Perhaps, however, you want to change the name of the EXPENSE.YTD file to a more descriptive extension with the same root name. You can specify the entire old name and an entire new name in the command line as in the previous example. An easier way, however, is to use an * wild card for the root file

name and change only the extension. You can see from examining the directory listing that only one file with the root file name of EXPENSE is available for RENAME to process with a wild card. You then issue the following command:

RENAME *.YTD *._89

The new file name for EXPENSE.YTD becomes EXPENSE._89. You can confirm that only the extension of the EXPENSE.YTD file was renamed by using DIR to list the directory:

```
 Volume in drive C is LAP TOP
 Directory of  C:\NAMES

.            <DIR>      8-29-89   9:29a
..           <DIR>      8-29-89   9:29a
SALES_08 REP     1664   8-01-89   2:51p
SALES_09 REP      128   9-23-89   1:34p
EXPENSE  _89    25372   9-23-89   8:00p
        5 File(s)   1976320 bytes free
```

While viewing this directory listing, suppose that you decide to change the names of the first two files to reflect that they are month-to-date reports. You want to keep the same extension for both files, so the extension specifier should be an * in both the old file name and also the new file name. You want the new file name for both files to take the form of `MTD_S_nn`, where `nn` is the month. Because both REP files have the two-number month as a unique set of characters in the current file name, you can use a combination of * and ? wild cards to form the old and new file parameters. You issue the command as follows:

RENAME *.REP MTD_????.*

The old file specifier will match both REP files in the current directory as candidates for renaming. If you specify **MTD_????**, the RENAME command selectively changes the first four characters of the old file names from SALE to MTD_ and keeps the next four characters—represented by the ? wild card—intact. The * in the extensions of the new files allows the new names to keep their old extensions. Using the DIR command to see another directory listing, you find the names changed as follows:

```
 Volume in drive C is LAP TOP
 Directory of  C:\NAMES

.            <DIR>      8-29-89   9:29a
..           <DIR>      8-29-89   9:29a
MTD_S_08 REP     1664   8-01-89   2:51p
MTD_S_09 REP      128   9-23-89   1:34p
EXPENSE  _89    25372   9-23-89   8:00p
        5 File(s)   1974272 bytes free
```

Another possible use for RENAME is in giving a file an unusual name. You recall that the space character is not allowed in file names on the command line. A space is a delimiter character that causes the command interpreter, COMMAND.COM, to misread the parameters on the command line. By using wild card ? characters in a RENAME command, it is possible to include spaces in a file name. If you don't want someone's prying eyes to read one of your files by using the TYPE command, the spaces can add the protection that you need.

RENAME can use the wild card to put spaces into a file name during a renaming session because the space character is not literally present in the command line. COMMAND.COM is not tripped up as it would be in a literal renaming using a space.

If, for example, you added the file NEW.REP to the default directory, the new file would appear in the output of a DIR command as follows:

```
 Volume in drive C is LAP TOP
 Directory of  C:\NAMES

.           <DIR>       8-29-89   9:29a
..          <DIR>       8-29-89   9:29a
MTD_S_08 REP      1664   8-01-89   2:51p
MTD_S_09 REP       128   9-23-89   1:34p
EXPENSE  _89     25372   9-23-89   8:00p
NEW      REP      1644  10-01-89   5:55p
        6 File(s)   1970176 bytes free
```

Because the root file-name provision in the directory entry is eight characters long, DOS pads the root file name NEW with five spaces in the directory entry. You can retain some of the space characters that DOS included in the NEW root file name by issuing the following command:

RENAME NEW.REP MTD?S?10.*

As always, DOS replaces the literal characters given in the new file-name parameter and retains the wild-card pattern that matches the old file name. In this case, the positions of the ? wild cards represent the spaces DOS put in the last five directory entry character positions. In the directory listing of this default directory, you can see that the spaces in the file name were added without objections from DOS.

```
 Volume in drive C is LAP TOP
 Directory of  C:\NAMES

.           <DIR>       8-29-89   9:29a
..          <DIR>       8-29-89   9:29a
MTD_S_08 REP      1664   8-01-89   2:51p
MTD_S_09 REP       128   9-23-89   1:34p
EXPENSE  _89     25372   9-23-89   8:00p
MTD S 10 REP      1664  10-01-89   5:55p
        6 File(s)   1968128 bytes free
```

Be aware that you cannot specify the file name **MTD S 10.REP** literally in a command. COMMAND.COM will not accept it at its face value. You can use wild cards to manipulate or erase a file name that contains spaces.

One common RENAME mistake is to include a wild-card pattern that attempts to rename more than one file to the same new name. For example, if you want to indicate by the files' names that the .REP extension files in this directory were no longer needed, you could use a wild card * to rename the REP files to the root file name of DONE. Because duplicate file names exist in the directory, however, this command will not work:

RENAME *.REP DONE.*

You immediately see the following message from DOS:

```
Duplicate file name or File not found
```

Although this message from DOS seems to indicate that the RENAME command has not been able to rename any REP files, the first REP file has been renamed. You can confirm this renaming by listing the directory with the DIR command:

```
 Volume in drive C is LAP TOP
 Directory of  C:\NAMES

.            <DIR>       8-29-89   9:29a
..           <DIR>       8-29-89   9:29a
DONE     REP      1664   8-01-89   2:51p
MTD_S_09 REP       128   9-23-89   1:34p
EXPENSE  _89     25372   5-10-89   8:00p
MTD S 10 REP      1664  10-01-89   5:55p
        6 File(s)   1968128 bytes free
```

Notice that the first .REP extension file in the directory has been successfully renamed to DONE.REP. RENAME issued the duplicate file message when attempting to rename the MTD_S_09.REP file. The resulting file name would have been a duplicate of the first REP file's new name; DOS will not allow two files called DONE.REP to be in the same directory.

Deleting Files with ERASE (DEL)

Cue:
The ERASE and DEL commands remove files from the disk.

The ERASE and DEL commands remove files from the disk. Both command names produce identical results; you can use ERASE and DEL interchangeably. When you erase a file with ERASE or DEL, the file is no longer accessible by DOS, and the file's directory entry and storage space is available to DOS for additional storage of another file. Because ERASE destroys files, use the command with caution. ERASE is a necessary file-management command, however. Unless you erase unwanted or unnecessary files from a disk, the disk can even-

tually reach full storage capacity. If the disk is a hard disk, you need to erase files in order to use the hard disk for your primary data storage.

Issuing the ERASE Command

You can use either form of this command—ERASE or DEL. The syntax for the command is as follows:

ERASE *d:path***filename.ext**/P*

DEL *d:path***filename.ext**/P*

d: is the optional drive containing the disk which holds the file(s) to be erased. If you omit the drive specifier, the logged drive is used.

path\ is the optional directory path to the file(s) to be erased. If you omit the path specifier, the current directory is assumed.

filename.ext is the specifier for the file(s) to be erased. Wild cards are allowed in the file name and extension. A file specifier or a path specifier is required.

The */P* switch is available with V4 and later versions. Using the optional /P switch causes ERASE to *pause* and display the name of each file to be erased before deleting the file. You answer Y to delete the file or N to leave the file intact.

Understanding the Operation of ERASE

When you issue an ERASE command, DOS locates the file's entry in the directory and marks the directory entry with a special internal indicator. DOS uses this indicator to search for available directory entries when a new file is being added to the directory. By reclaiming a deleted file's directory entry, DOS can control the expansion of a subdirectory or reclaim one of the limited root directory entries.

ERASE does not affect the contents of the file's allocated clusters. ERASE does not "record over" the file's data as erasing a cassette tape records over existing audio. The entire operation of ERASE involves altering DOS's bookkeeping records in the directory and the file allocation table (FAT). The directory entry for the file receives its special indicator, and the FAT cluster chain for the file is deallocated. DOS marks the file's clusters as being "free."

Until another file is added to the directory or another file in any directory is expanded or added, the DOS bookkeeping records for the deleted file remain relatively intact. DOS will not, however, reverse the effect of erasing a file. To DOS, the file is gone even though the file's data is temporarily intact.

Reminder:
Keep proper backup files to ensure protection against the accidental erasure of a file.

Some special utilities available from third party companies can "unerase" a file. These unerase utilities take advantage of DOS's internal file-erase methods by "fixing" the erased file's directory entry and reconstructing the erased file's cluster chain. If another file has been added, however, these unerase utilities cannot recover the erased file. If you are interested in an unerase utility, check with your dealer for a recommendation. Just remember that the unerase utilities must be used immediately if they are to work. Keeping proper backup files is the best insurance against the accidental erasure of a file.

Understanding the General Rules of ERASE

- ERASE will not erase files marked with the read-only attribute.
- ERASE will not erase a directory file, volume label, or a hidden or system file.
- If you type **DEL subdirectory name**, DOS tries to delete all the files in **subdirectory name**.
- When a file is erased, you cannot access the file through DOS. The file is effectively removed from the disk.
- Unless you use the /P switch (V4 only), ERASE silently erases a file and issues no messages.
- If you use the full wild-card file specifier *.*, DOS prompts as follows:

  ```
  Are you sure (Y/N)?
  ```

  You answer Y to delete all erasable files in the directory or N to cancel the command. Files marked read-only will not be erased.

Using ERASE with the ATTRIB Command

Suppose that in the process of working in dBASE, you add the file PROG.BAT to the dBASE directory. Now you no longer need the file. You can erase the PROG.BAT file while in any directory on the same disk by issuing the following command:

ERASE \DBASE\PROG.BAT

DOS erases the PROG.BAT file without issuing a message. You can erase PROG.BAT from \DBASE as your current directory with the following command:

ERASE PROG.BAT

In the course of working with dBASE, you add several files. These added files are not part of the core dBASE files that were marked as read-only in the

ATTRIB example you saw in previous sections of this chapter. The added files have various root file names as well as several extensions. You list the \DBASE directory and see the listing in figure 7.2.

```
C:\DBASE>DIR /W

 Volume in drive C is LAP TOP
 Directory of  C:\DBASE

.            ..           DTL          ENGINE       SAMPLES
SQLHOME      DBSETUP  EXE DBSETUP  RES DBSETUP  PRD DBSETUP  OVL
DBASE3   OVL DBASE    EXE GENERIC  PR2 ASCII    PR2 FX80_1   PR2
CONFIG   DB  DBASE1   RES DBASE6   OVL DBASE1   OVL DBASE3   RES
DBASE2   OVL DBASE1   HLP DBASE5   OVL DBASE4   OVL DBASE2   HLP
DBASE2   RES DBLINK   EXE LABEL    GEN REPORT   GEN PROTECT  OVL
FORM     GEN MENU     GEN DCONVERT EXE LABEL    PRF REPORT   PRF
DBLINK   RES QUICKAPP GEN DOCUMENT GEN RUNTIME  EXE RUNTIME  RES
RUNTIME1 OVL RUNTIME3 OVL RUNTIME2 OVL RPROTECT OVL RUNTIME4 OVL
BUILD    COM BUILDX   EXE BUILDX   RES DBSAMPLE BAT MENGEN   DBO
CATALOG  CAT UNTITLED CAT SCRGEN   PRG MENUT    PRG MENENG   PRG
MENGEN   PRG BLUE     EXE CARLTON  SET DWS      MEM CARLTON  BAT
      60 File(s)   1091584 bytes free

C:\DBASE>
```

Fig. 7.2.

A directory listing for dBASE program files.

The listing in figure 7.2 is a typical example of a directory reserved for an applications program. As you notice, the directory contains a large number of files, yet some files are not part of the program's operation files.

Suppose that you want to erase the recently added files from the directory and leave only the original core dBASE files. If you have marked the core files as read-only (as in the example given with the ATTRIB command), you can erase the added files at this time because you are finished with the files and no longer need them on your hard disk.

You can use the directory listing to erase the unneeded files one at a time, or you can try to isolate common extensions to use as a wild-card pattern. In this case, the easiest way to erase the added files is to use the full wild-card file parameter. The core dBASE files will not be erased because they were previously marked as read-only. Only the added files will be erased. Issue the following command:

ERASE *.*

DOS sees the full wild-card file specifier and prompts for confirmation with the following:

```
Are you sure (Y/N)?
```

You respond with Y; the added files are deleted. Using the ERASE command in conjunction with the ATTRIB command makes erasing a random group of files simple.

Viewing a File's Contents with TYPE

When you need to see the contents of a text file, you can use the TYPE command. TYPE gets its input from a file (or a device) and sends the ASCII interpretation of the file's contents to the screen. You can use the MORE filter with TYPE to keep the file's text from scrolling off the screen as you view the file. You can also redirect the output of TYPE to your printer to produce a hard copy of the file.

Cue:
The TYPE command produces screen output of any character-based file.

Because TYPE enables you to see the contents of a file, TYPE is an important file-management command. When you are cleaning up a directory or a diskette, you can use TYPE to refresh your memory about files whose names don't "ring a bell." When you are configuring your system, you can use TYPE to see the contents of your CONFIG.SYS and AUTOEXEC.BAT files. When you are installing a new applications program, you can use TYPE to see the latest instructions in the package's README.DOC or (equivalent) file. TYPE produces screen output of any character-based file.

Issuing the TYPE Command

The syntax for the TYPE command is as follows:

TYPE *d:path***filename.ext**

d: is the optional drive that holds the disk containing the file to be typed. If you omit the drive, the default drive is assumed.

path\ is the optional path to the file to be typed. If you omit the path, the default directory is assumed.

filename.ext is the name of the file to be typed. You must name the file in the command line. Wild cards are not permitted.

Understanding the Operation of TYPE

TYPE is a character-based command because it outputs a file's contents to the screen. TYPE is designed to display the content of a text file that contains ASCII

characters. When you issue a TYPE command, DOS opens the specified file and sends the file's content to the output device (normally the screen) as a stream of bytes. The bytes are displayed as ASCII characters. When DOS encounters a Ctrl-Z, end-of-file character, the input file is closed and the output terminates.

If you use DOS redirection to send the output stream to a printer, the printer may store characters faster than it can print the character stream. As a consequence of the printer's buffering of yet-to-be-printed characters, Ctrl-C or Ctrl-Break will not stop the printer immediately.

Your computer screen will wrap a text-line output produced by TYPE to the following line if the number of characters in the line exceed 80. The same line, when redirected to a printer, may not wrap to the next print line after 80 characters if the printer is capable of printing a line in excess of 80 characters. In such a circumstance, the printer-redirected output of TYPE may print off the page. TYPE is faithfully sending the output stream with the stream's embedded line feed and carriage return characters, but the line's format doesn't match the printer's paper width.

With a binary file, the screen or redirected printer output produced by TYPE will likely contain nonsense characters that are the result of non-character bytes being displayed as characters. Control codes embedded in the output stream will produce unexpected results. Usually, using TYPE to display the contents of a binary file on the screen will not cause a problem. Using TYPE with the output redirected to a printer, however, can produce undesirable (although not destructive) results.

Reminder:
Using TYPE to send the output of a binary file to the printer may cause unexpected results.

A file containing byte values of an ASCII form-feed character can cause a printer to feed many sheets of paper through the printer before you can stop TYPE's output with Ctrl-C or Ctrl-Break. Binary sequences in a file can cause the printer to change fonts, line spacing, graphics mode, and other printer-specific functions. If you send binary output to your printer with TYPE, you may have to turn the printer off and then back on to clear any operational controls changed by the redirected output of TYPE. You may have to stop the TYPE command before turning the printer back on.

Understanding the General Rules of TYPE

- Because TYPE is a character-based command, the output of TYPE stops when the first Ctrl-Z is encountered in the file being sent to a device as output. File bytes following the first Ctrl-Z will not be sent.
- You must give TYPE a literal file name. TYPE does not not accept wild cards in the file specifier. TYPE is therefore limited to a one-file-at-a-time use.

- TYPE interprets any file as a text file. If you use as the file specifier a binary file, such as a .COM or .EXE file, TYPE's output may produce graphical characters, control-character sequences, and beeps, and may hang your computer. A "hung" computer will not respond to keyboard input, and you will have to reboot.
- You can pause the output of TYPE by pressing Ctrl-S or Ctrl-Num Lock.
- You can terminate the output of TYPE by pressing Ctrl-C or Ctrl-Break.

Viewing the Contents of CONFIG.SYS with TYPE

To view the contents of a short text file, you can give a simple form of the TYPE command. In this example, suppose that you want to view the contents of your CONFIG.SYS file. Issue the following command:

TYPE \CONFIG.SYS

DOS sends the character content of CONFIG.SYS to the screen for you to view:

```
BUFFERS=15
FILES=15
DEVICE=\DOS\DRIVERS\EMM.SYS
DEVICE=\DOS\DRIVERS\ANSI.SYS
```

Using TYPE and Redirection

Cue:
You can use the MORE filter to control TYPE's output to the screen.

Many text files are too large to fit on the display at one time. You can control TYPE's output to the screen by using the MORE filter. To view a 30-line README.TXT file, issue the following command:

TYPE README.TXT | MORE

DOS types the first screen of text from the README.TXT file and displays the following prompt:

```
-- More --
```

When you press a key, TYPE will display the rest of the README.TXT file's content.

If you want a simple printed copy of a text file's content, you can redirect the output of the TYPE command to the printer. Because TYPE does not format the text into pages, the printed output may not "break" at page boundaries. To

print the contents of README.TXT, ensure that your printer is on-line and issue the following command:

TYPE README.TXT >PRN

The file is then printed by your printer. You can also redirect TYPE to another device such as .COM1 or .COM2 (see Chapter 4 on redirection).

Comparing Files with COMP

Cue:
The COMP command is available with some versions of DOS.

The external command COMP compares two files or two sets of files to find differences in the files. Any differences are then reported on-screen. COMP is supplied with PC DOS V2.0 and later versions and with MS-DOS V3.3 and later versions. Check your version of DOS to determine if you have the COMP command. MS-DOS includes a similar command, FC, which is discussed in the next section.

File comparison is a very useful file-management tool. Unlike the DISKCOMP disk-level command, COMP reports differences at the file level. With COMP you can determine any of these conclusions about compared files:

- The files are identical in content and identical in size.
- The files are identical in size, but different in content.
- The files are different in size and content.

In a practical sense, COMP is useful for determining if a file is the same as another file on a different disk or in a different directory. If the compared files have the same name, you can easily copy an older version of the named file to another disk while not realizing that the newer version in another directory is different. You might copy a file to a file name such as TEMP to have a temporary copy. After some time, you might forget which file TEMP was copied from. Using COMP with TEMP and other files will show the file that matches TEMP.

Issuing the COMP Command

The syntax for the COMP command is as follows:

COMP *d1:path1\ filename1.ext1 d2:path2\filename2.ext2*

d1: is the optional drive that contains the first file or set of files (the primary set) to be compared. If you omit the *d1:* specifier, the current drive is assumed.

path1 is the optional path to the file(s) to be compared. If you omit the *path1* specifier, the default directory is assumed.

filename1.ext1 is the optional file specifier for the file to be used as the basis of comparison. Wild cards are allowed in both the file name and extension. If you

omit the file specifier but include a drive or a path, COMP assumes that the file specifier is *.*.

d2: is the optional drive specifier which has the second file or set of files (the secondary set) to be used in the comparison. If you omit the d2: specifier, but include a path or file-name specifier, the logged disk is assumed.

path2\ is the optional path to the secondary set of files. If you omit the path2\ specifier, COMP assumes the default directory.

filename2.ext2 is the optional file name and extension of the secondary set of files. Wild cards are allowed in the file name and extension. If you omit the secondary set file-name specifier, COMP assumes that the secondary set files are the same name as the primary set files.

If you issue COMP with no parameters, DOS prompts you for parameters. Use COMP alone when you need to change disks after the external COMP command is loaded from disk.

Understanding the Operation of COMP

COMP reads one file and compares each byte to another file. Bytes are compared based on their relative position in the file. COMP is not aware if the bytes represent ASCII-character data or binary data. COMP does check both files for the Ctrl-Z end-of-file markers. If an end-of-file marker is not found, COMP displays the message `EOF mark not found`. The end-of-file marker message is informational only and not an indication of a problem.

Once the COMP program is loaded into memory, you can continue to do file comparisons by answering Y to the prompt `Compare more files (Y/N)?` COMP will then prompt you for parameters.

You can redirect the output of COMP to the printer, but the final prompt of a comparison, `Compare more files (Y/N)?` as well as prompts for parameters, will be redirected to the printer. If you use redirection, you need to read the prompts from the redirection device. When you redirect the output of COMP to a file, you will have to anticipate any prompting messages because you will have no way to see prompts.

If you use wild cards in the secondary set file specifier, the characters used for the pattern match in place of wild-card characters come from the first set file name, not from a directory search of the second set directory names.

The command **COMP C:*.* A:*.BAT**, for example, compares all batch files in drive A: to those in the current directory of drive C:. The contents of drive A: are shown in a directory listing as follows:

```
 Volume in drive A is MINIFLOP
 Volume Serial Number is 0000-0000
 Directory of  A:\

DOSSHELL BAT       196 09-11-89   4:20p
FORMAT   COM     22923 06-17-88  12:00p
       2 File(s)     705536 bytes free
```

The BAT extension file in the current directory of C: appears in the output of a **DIR C:*.BAT** command as follows:

```
 Volume in drive C is MY DISK
 Directory of  C:\DOS4

DOSSHELL BAT       196 09-11-89   4:20p
       1 File(s)    3305472 bytes free
```

The result of the COMP command is as follows:

```
C:DOSSHELL.BAT and A:DOSSHELL.BAT

EOF mark not found
Files compare OK
```

As you see, only the DOSSHELL.BAT file of the two disks was compared. The *.* wild card file specifier for the second set of files did not include the A:FORMAT.COM file.

Understanding the General Rules of COMP

- If you do not specify parameters, COMP will first prompt for the primary set specifier and then for the secondary set specifier.
- As each file in the set is compared, the file-compared names are displayed.
- If the two files being compared are different sizes, the only report issued for those files by COMP is `Files are different sizes`. No further comparison of the two files is made.
- If COMP detects differences in two files of the same size, the differences are reported to the screen as byte offsets (positions in the file) and values of the differing bytes. Byte offsets and values are given in hexadecimal (base 16) notation.
- COMP reports the first 10 differences in the two files and then reports `10 Mismatches - ending compare`.
- When all sets of files are compared, COMP prompts for additional comparisons.

Comparing Two Text Files with COMP

Suppose that two text files identical in size, ORIGINAL.TXT and ANOTHER.TXT, are located in the default directory of the logged drive A:. By using the TYPE command, you can see the contents of the files to obtain a gauge of how they will compare. (Normally, you would compare the files first to be more efficient.) When you view the output of the two TYPE commands, you see the following for the ORIGINAL.TXT file:

```
This file contains the following numbers:
1
2
3
4
5
6
7
8
9
```

For the file ANOTHER.TXT, you see the following:

```
This file contains the following numbers:
9
8
7
6
5
4
3
2
1
```

To compare the two files, issue the following command:

COMP ORIGINAL.TXT ANOTHER.TXT

COMP reports as follows:

```
A:ORIGINAL.TXT and A:ANOTHER.TXT

Compare error at offset <beep>21
File 1 = 31
File 2 = 39

Compare error at offset <beep>24
File 1 = 32
File 2 = 38
```

```
Compare error at offset <beep>27
File 1 = 33
File 2 = 37

Compare error at offset <beep>2a
File 1 = 34
File 2 = 36

Compare error at offset <beep>30
File 1 = 36
File 2 = 34

Compare error at offset <beep>33
File 1 = 37
File 2 = 33

Compare error at offset <beep>36
File 1 = 38
File 2 = 32

Compare error at offset <beep>39
File 1 = 39
File 2 = 31

EOF mark not found
Compare more files (Y/N)?
```

In this output, the <beep> indicator is a substitute for the PC's beep (bell) signal. COMP found 8 mismatches in the files. Because the number sequences in the files are reversed, you might expect to see 9 mismatches reported. Remember, however, that the number 5 is in the same position in both files.

Notice that COMP detected that the end-of-file marker is missing in the files. Although these files are ASCII text files, they do not contain the Ctrl-Z.

If you attempt to use wild cards in file parameters to eliminate command-line keystrokes, the result can be different than you might expect. Issuing the following command, for example, would seem to save keystrokes:

COMP ORIG*.TXT ANO*.TXT

Remember that wild cards in the secondary set file parameters are filled in from the primary set file parameter. The preceding command results in this comparison:

```
A:ORIGINAL.TXT and A:ANOGINAL.TXT

A:ANOGINAL.TXT - File not found

Compare more files (Y/N)?
```

Use wild cards in the second set file parameter only if the second set files will match the first set files.

Performing a Full-File Comparison with FC

Reminder:
The FC command is not available with all versions of DOS.

The external FC command is similar to the COMP command in that both compare files. FC, however, is a more capable command than COMP because FC can report ASCII text differences in readable text form. FC also does binary file comparisons. FC is available with some versions of MS-DOS. FC is not available with PC DOS.

Issuing the FC Command

The FC command has two general syntax forms. One form allows the /B switch for a forced *binary* comparison. The other form allows the remaining switches in an ASCII comparison. The two forms of syntax are as follows:

FC */B d1:path1***filename1.ext1** *d2:path2***filename2.ext2**

or

FC */A/C/L/LB*n/*N*/nnnn/*T*/*W d1:path1***filename1.ext1**
*d2:**path2***filename2.ext2**

NOTE: All switches *must* precede the other parameters.

d1: is the optional drive containing the first file to be compared. If the first drive specifier is omitted, the default drive is assumed.

path1\ is the optional path of the directory containing the first file. If the first file's path is omitted, the default directory is assumed.

filename1.ext1 is the file name of the first file. A file name must be entered. Wild-card characters are *not* allowed.

d2: is the optional drive containing the second file to be compared.

If the second drive specifier is omitted, the default drive is assumed.

path2\ is the optional path of the directory containing the second file. If the second file's path is omitted, the default directory is assumed.

filename2.ext2 is the file name of the second file. A file name must be entered. Wild-card characters are *not* allowed.

The switches are explained more fully in the next section.

Understanding the Operation of FC

FC works in two modes. FC compares two files in the ASCII mode or compares two files in the binary mode. Each mode reports its own formatted output. FC defaults to ASCII mode comparison when the files to be compared do not have .EXE, .COM, .SYS, .OBJ, .LIB, or .BIN extensions. In ASCII mode, FC compares two files on a line-by-line basis. Lines from both files are held internally in a line buffer. FC uses the lines in the buffer to compare the first file to the second.

Reminder:
FC can compare files in ASCII mode or in binary mode and produce readable output.

If a difference is detected by FC, the first file's name is displayed, followed by the last matching line from the first file. The first mismatching line and subsequent mismatching lines from the first file are then displayed. FC then produces output that contains the first matching line from file 1 that synchronizes the matching of file 2.

After displaying mismatch information about file 1, FC then repeats the same sequence for file 2. The file 2 name is displayed first, followed by the last matching line to file 1, followed by the mismatched lines to file 1, and ending with the first resynchronizing line of the two files.

The result of this output is a tool to identify the point-of-difference detection, the difference itself, and the point that the detected difference ended. You use this output as a short-cut alternative to a side-by-side comparison of the files' contents.

In the ASCII mode, FC offers several switches to modify the operation or output format of FC. Table 7.3 describes the action of each switch available in ASCII mode.

Table 7.3
FC ASCII Mode Switches

Switch	*Action*
/A	Instructs FC to *abbreviate* its output (V3.2 and later versions).
/C	Instructs FC to ignore the *case* of alphabetic characters.
/L	Instructs FC to do a *line-by-line* comparison on files that have .EXE, .COM, .SYS, .OBJ, .LIB, or .BIN extensions (V3.2 and later versions).
/LBn	Sets the number of *lines* in FC's *buffer* to *n*. The default number is 100 (V3.2 and later versions).
/N	Instructs FC to include the line *numbers* of lines reported in the output (V3.2 and later versions).
/nnnn	Establishes as *nnnn* the number of lines that must match after a difference to resynchronize FC.

Table 7.3—*continued*

Switch	*Action*
/T	Instructs FC to view *tab* characters as literal characters rather than tab-expanded spaces (V3.2 and later versions).
/W	Instructs FC to ignore tabs, empty lines, and spaces.

In binary mode, FC compares two files byte for byte. At the first difference, the byte offset position in the first file is reported along with the value of the two files' bytes at the position. The offset as well as the byte values are reported in hexadecimal (base 16) form.

In binary mode, FC does not attempt to resynchronize the two files by finding an adjusted point of byte agreement. If one file has an additional byte at one offset only, not only will FC report the difference, but all subsequent bytes of the file will likely be reported.

If one file is longer than its comparison file, the binary mode will compare as many bytes as are present and then report that one file is longer. When a binary file comparison results in a long listing of differences, you may want to stop the FC operation with Ctrl-C.

Only one switch is available in the binary mode. The /B switch causes the comparison to be binary even if file extensions indicate that the files are not binary. You use the /B switch to compare two text files in binary mode. You may find situations in which you would rather have the binary mode output format of FC than the ASCII mode format. The binary mode format reports differences as pairs of hexadecimal values. You can then see the values of characters, such as Ctrl-G (bell), that do not produce printed output.

Understanding the General Rules of FC

- When the /B (binary mode) switch is used, including /L, /N, and /nnnn produces an error message from DOS. All other switches included with /B are ignored by FC.
- The internal buffer defaults to 100 lines for comparison, but can be changed with the /LBn switch.
- The default number of lines that must match in an ASCII comparison after a difference has ended is 2. The files are then considered to be resynchronized. The number of "must match" lines can be changed using the /nnnn switch by setting nnnn to the desired value.
- Unlike the COMP command, FC requires file-name specifiers consisting of literal file names. Wild cards are not allowed.

Using FC To Compare Two Versions of a Memo

Suppose that the file C:WP\REPLY.MEM is a text file containing a brief memo. The content of the file is as follows:

> Dear Mr. Smith,
>
> I want to thank you for considering me for the position of Senior Buyer at Smith Industries. I have spent much time deciding between staying at Aberle Manufacturing and joining your firm. I hope you appreciate that my decision was not easy.
>
> I have decided to join Smith Industries. I feel that my decision is the best for my career.
>
> Again, thank you for your consideration.
>
> Sincerely,
>
> Megan Sue Heichelbech

The originator of the memo, Megan Sue Heichelbech, revised the memo and saved a copy to the file A:\REPLY.MEM. At a later time, Ms. Heichelbech elected to send the memo, but first compared the two files with the same name by issuing the following command:

FC /N/W REPLY.MEM A:REPLY.MEM

FC reported as follows:

```
***** reply.mem
    5:  your firm. I hope you appreciate that my decision was not easy.
    6:  I have decided to join Smith Industries. I feel that my decision
    7:  is the best for my career.
***** a:reply.mem
    5:  your firm. I hope you appreciate that my decision was not easy.
    6:  I have decided to decline your offer. I feel that my decision
    7:  is the best for my career.
*****
```

FC found a difference in line 6 of the two files. The last matching line was line 5, and the next matching line after the difference (where FC resynchronized the comparison) was line 7. FC ignored blank lines because the /W switch was included. FC added line numbers because the /N switch was included.

Tapping the Power of COPY

Cue:
The DOS COPY command has the capability to copy files as well as copy to and from logical devices.

The COPY command is the DOS workhorse. Copying files is a fundamental job for disk operating systems. DOS provides COPY with the capability to copy files as well as the capability to copy to and from logical devices. Figure 7.3 shows the COPY command used with a variety of possible inputs and two possible outputs. Three of the possible inputs consist of more than one file or logical device. DOS can join two or more inputs into one output in a process called *concatenation*.

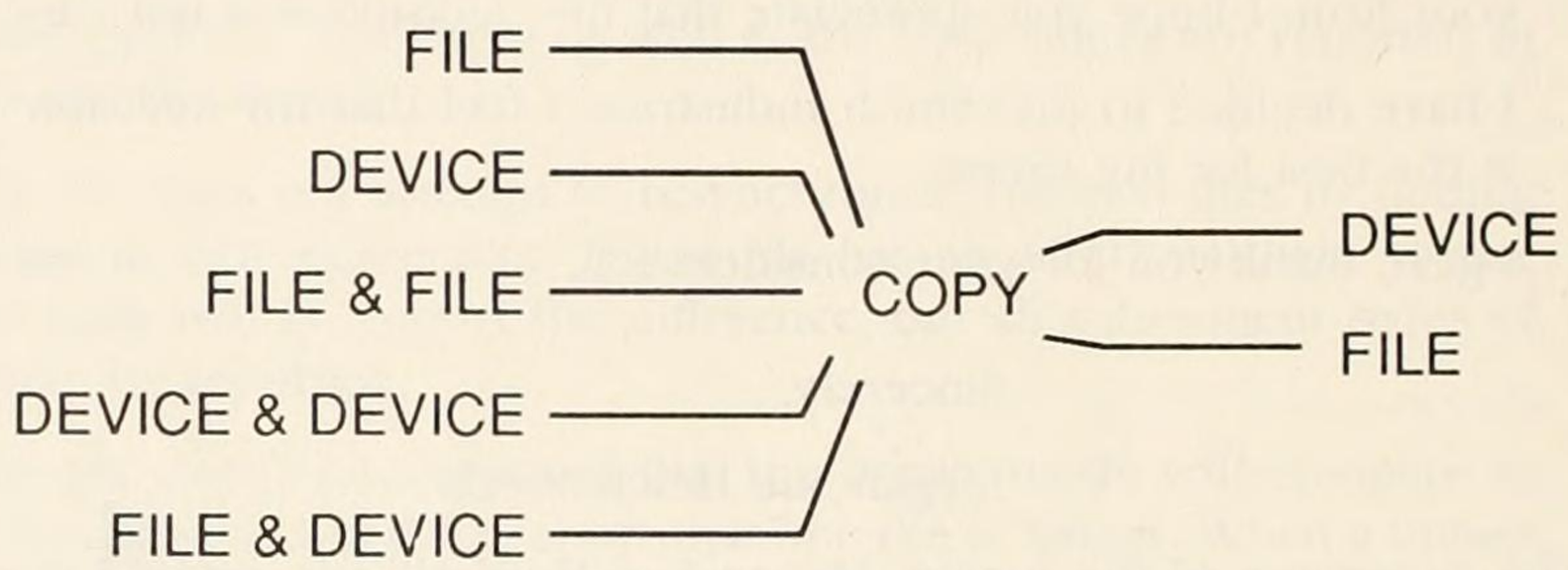


Fig. 7.3
The possible inputs and outputs of the COPY command.

Cue:
When COPY concatenates two files, the files are joined in the output file.

When COPY concatenates two files, the files are joined in the output file. Concatenation is not new in disk operating systems. The UNIX operating system uses the command CAT, for concatenate, to copy files and devices. The DOS COPY command can concatenate a device's output and a file, a file and a file, or even a device's output to another device's output. The resulting output can go to a file or a device. COPY is versatile when accepting inputs and providing outputs.

COPY can selectively use either ASCII or binary mode for its inputs and outputs. Switches given with the command control COPY's mode selection. In ASCII mode, COPY interprets a ^Z (end-of-file marker) as the logical end of file and considers the current input file or device as finished. In ASCII mode, COPY places a ^Z as the last character in the output file or device data stream. Applications programs and DOS commands that use the ^Z end-of-file marker are accommodated by COPY.

The COPY command accepts wild cards in file specifiers to provide a versatile syntax. With drive and path parameters, COPY can reach any user file on a system from any logged drive and current directory. With *.* file name parameters, COPY can take every file from a fragmented diskette and copy each file to a newly formatted diskette. By doing so, COPY eliminates the fragmentation on the second diskette.

Even with the presence of the related commands DISKCOPY and XCOPY, COPY is the most widely used file system command. If you are not taking full advantage of COPY's range of features, this discussion of COPY will show you how.

Issuing the COPY Command

In a symbolic representation, the COPY command looks like this:

COPY *from this source or these sources to this destination*

COPY always orders the parameter requirements in a direction of source to destination or target. When you include the mode of reading and writing the files (or devices), COPY symbolically looks like this:

COPY *from source in this mode to destination in this mode*

The symbolic mode for this example is either ASCII or binary for both the source and also the destination.

If you incorporate the symbolic view of COPY into syntax lines, you see the following syntax forms:

COPY *d1:path1***filename1.ext1** */A/B*
*d0:path0***filename0.ext0** */A/B/V*

or

COPY *d1:path1***filename1.ext1** */A/B* +
*d2:path2***filename2.ext2** */A/B* +. . .
*d0:path0***filename0.ext0** */A/B/V*

d1: and *d2:* are the optional source disk drive specifiers. If you omit a source disk specifier, the logged drive is assumed.

path1\ and *path2*\ are optional source path specifiers. If you omit a source path specifier, the current directory is assumed.

The + character delimits source files that are to be concatenated.

filename1.ext1 and **filename2.ext2** are source file-name and extension specifiers. You must give a source file specifier.

The . . . signifies that other drive, path, and source file name parameters may be included in a concatenation.

The optional */A* switch invokes *ASCII* mode and affects the file specified directly preceding the switch and all files following the switch up to the file preceding a subsequent /B switch.

The optional /*B* switch invokes *binary* mode and affects the file specified directly preceding the switch and all files following the switch up to the file that precedes a subsequent /A switch.

The optional /*V* switch causes COPY to perform a read-after-write *verification* of the output sectors of the destination file. The destination file is not compared to the source file(s) directly. Instead, the memory-resident working buffer portion of the file being written is compared to the same data that is read back from the destination file after being written to the file by COPY. The result of this memory-to-file comparison is an effective file-data validity check.

Understanding the Operation of COPY

Cue: *COPY's operation proceeds from source to destination.*

COPY approaches its work serially; the command's operations occur one step at a time. The flow of COPY proceeds from source to destination. Internally, DOS assures that affected directories and FATs are updated correctly. COPY's initial act is determining the first source file or device.

If the first source has a reserved device name, COPY asserts the ASCII mode, reports the device's name on-screen, and awaits input from the device. If the first source is not a device, COPY assumes that the source is a file. Unless a previous /A switch, an associated /A switch, or an associated + operator is present, COPY asserts the binary mode. If the current source is a file, COPY determines the full path specification for the source file.

COPY scans the source parameter for the first drive, path, and file name. If any of these specifiers is missing, COPY consults DOS's internal log for current settings to use in the parameter. When it finds the intended directory of the source file(s), COPY reads the source directory from the top entry downward to find the file or a file that matches a wild-card pattern. When the file is located, COPY reports the file name on-screen and begins to read the file.

Data that is read from the input file or device is temporarily stored in a RAM buffer which fills during the input to COPY. When the buffer first fills or the end of the source file is encountered, COPY establishes the destination parameter.

If the destination is the device, COPY begins to write the contents of the buffer to the device until the buffer is empty. COPY then reads more data from the source, and the cycle repeats until all data is copied from the source to the destination device.

Reminder: *COPY uses default values if drive and path specifiers are omitted.*

If the destination is a file, COPY establishes the destination file's directory. COPY uses the given destination drive and path specifiers to locate the destination directory. Again, COPY will use default values if drive and path specifiers are omitted. The destination directory is searched from the top downward using the name of the destination file. If the destination file-name specifier con-

tains wild cards, COPY uses the characters from the source file to match the wild-card patterns in forming the destination file specification.

If COPY finds a duplicate full file specification as the destination parameter (same drive, directory, and full file name), COPY determines whether the operation is a concatenation or a straight copy. If a + operator is given in the source parameter or if a wild card is given in the source file name while a literal destination file name is given, COPY assumes that a concatenation operation is taking place. Otherwise, COPY assumes that a straight file-to-file copy operation is taking place. If a concatenation operation is taking place and a duplicate source/destination file name is not the first source file name, COPY issues a warning message that the contents of the destination file are lost before the copy operation executes. If the operation is a straight file-to-file copy, COPY issues a message that the file cannot be copied to itself and then aborts the operation.

When COPY finds the destination file's entry in the destination directory during a concatenation operation, or if the destination file name is different from the source file name, COPY truncates the current file and writes the contents of the buffer as the new contents of the truncated file. COPY retains the same directory entry and name for the new contents of the truncated file.

When COPY finds that the destination file is a new file (not currently in the destination directory), COPY establishes an entry in the destination directory for the new file and begins to write the temporarily buffered source data to space allocated for the new file.

If the /V switch is given in the command, sectors that are written to the destination file are read back from the new file and compared to the data in the buffer for verification. If the original buffer information and the read-after-write information do not agree, COPY issues an error message and aborts the operation.

COPY continues reading source information into the buffer and writing the full buffer to the destination until the source file is completely read to the end of file. If the destination file is affected by the /A switch, COPY appends the Ctrl-Z character to the end of the destination file. COPY then closes the destination file and updates the directory.

If the source parameter contains wild cards or the + concatenation operator, COPY repeats the entire process by moving to the next source for the copy.

If there are no other source parameters for COPY to process, COPY issues a message indicating the number of destination files and/or devices processed. With its job complete, COPY returns control to COMMAND.COM.

Understanding the General Rules of Copy

- The source parameter must contain at least one of the following: a drive specifier, a path specifier, a file specifier, or a device specifier.
- If the source-file specifier is omitted, all files in the specified (or default) directory of the specified (or default) drive are copied. This situation is equivalent to supplying the *.* source file specifier.
- Additional source parameters can be specified using the + operator for concatenation.
- If the source-file parameter contains a wild card and the destination parameter is a literal file name, the destination file will be the concatenation of source files matching the pattern of the wild card.
- If COPY detects an attempt to copy a single source file to itself (same drive, directory, file name, and extension), the copy operation is aborted.
- When a device is used as a source, no drive or path specifiers should be used in the source specification.
- When multiple source files are concatenated to a destination file that is one of the source files, the contents of the destination file will be its own source file at some point in the copy operation. Unless the destination file is specified as the first source file, the contents of the doubly specified source file will be lost before the copy is made. DOS will produce an error message to this effect.
- The optional destination parameter consists of a combination of drive, path, and file-name specifiers. An omitted drive or path specifier assumes the value of the default drive or directory. An omitted file-name specifier assumes the value of the source file-name specifier.
- COPY uses the ASCII mode by default for multiple source files (concatenation), device sources, and device destinations. All other operations use the binary mode by default. The /B operator is not allowed for source devices and will produce an error message. Source device input must end with the Ctrl-Z (F6 function key).
- The /A (ASCII) and /B (binary) switches affect the file specified directly preceding them and also affect subsequent files specified up to and not including the file specification which is directly followed by another /A or /B switch.
- When the /A switch is applied to a source file, the file is copied up to, but not including, the end-of-file marker (Ctrl-Z) or the physical end of file (as indicated by the byte count in the directory), whichever comes first.

- When the /A switch is applied to the destination file, COPY appends the Ctrl-Z character as the last character in the file.
- When the /B switch is applied to a source file, all bytes in the file, including Ctrl-Z characters, are copied to the destination file.
- When the /B switch is applied to a destination file, COPY does not append its own Ctrl-Z character to the destination file.

Respecting the COPY Command

COPY is certainly a versatile file-management workhorse. Yet even with its versatility, an errant COPY command can do nearly as much damage as an errant ERASE command, so be sure to treat COPY with respect. Many applications programs provide warnings regarding their internal file-copying commands. The DOS COPY command does not warn when it is about to overwrite existing files; it silently overwrites the files. If you confuse the source and destination when you are copying a file to a standby floppy, you can easily copy yesterday's standby file over today's work. You might not even notice the error until you need today's work. Not all mistakes with COPY, however, overwrite current data. Some mistakes just create file-management messes.

Relying on default drive and directory values when using COPY is convenient. Yet, if your current drive or directory isn't what you think it is, you can copy many files to the wrong disk or directory with one COPY command. The aftermath of cleaning the misplaced files from the erroneous directory or disk and getting the files into the correct directory or disk can be a tedious process. If the misplaced files have no common root name or extension that differs from the ones of the proper file residence of the disk or directory, you will need to copy these errant files one by one to their correct home.

Reminder:
For complete control of COPY, give the command drive, path, and file-name specifiers for both source and destination files.

To avoid sending files to the wrong directory, check the name of the logged disk or directory. Don't forget to check the name of the current directory on other drives on your system as well. You might have forgotten that you changed floppies before a coffee break. Use the DIR command with drive and path parameters for the source and destination of your next COPY command. Viewing the directories will take a bit longer, but not as long as copying misplaced files one by one. For complete control of COPY, give the command drive, path, and file-name specifiers for both source and destination files.

Watch out for wild cards as well. You can select a source wild-card pattern that results in the inadvertent overwriting of destination files. For example, suppose that you have three reports and one graph in the files REPORT1.TXT, REPORT2.TXT, REPORT3.TXT, and REPORT3.PIX. You decide to save a

Caution:
Be careful when using wild cards with COPY. You can inadvertently overwrite destination files.

standby copy of each file using a .SAV extension and issue the following command:

COPY REPORT?.* *.SAV

You intend for the ? wild card to differentiate the files by number after they are copied. The error in this reasoning is that both REPORT3.TXT and REPORT3.PIX will fit this source wild-card pattern and that both will be written to a file named REPORT3.SAV. The REPORT3.PIX copy will overwrite the existing REPORT3.SAV created as a standby copy of REPORT3.TXT. As a result, REPORT3.TXT will no longer have an equivalent standby copy. If you erase the .TXT extension files because you think that their contents are covered by the .SAV extension files, you will lose the information represented by REPORT3.TXT.

Many COPY problems stem from the way a computer user develops patterns or habits when entering commands. If you are accustomed to copying a file from the default directory of your hard disk to the diskette in drive A:, you're not unlikely to develop a rhythm of entering the COPY parameters. When a different COPY situation arises, the conditioning of the previous rhythm may cause you to enter the accustomed parameters and rely on the accustomed defaults. Without thinking, you may press Enter and rapidly overwrite your new data with old data. Perhaps you pause before pressing Enter and review the command line for accuracy. If you do, you're to be congratulated. But if you get hasty from time to time, you might destroy some data with COPY.

Using COPY in a Variety of Operations

To get a full view of COPY, this section presents straight file-to-file COPY commands as well as COPY commands that are concatenation operations. COPY commands that include devices as source and destination parameters are included. Use these examples as templates for your own COPY needs and watch out for included traps.

Copying from Directory to Directory

With today's hard-disk based PCs, you can find file management difficult if you avoid directory-to-directory COPY commands. This set of examples shows three command lines that copy a file from one directory to another while giving the file a new destination name. Each command takes full advantage of DOS defaults by omitting current values in the command. The source file's full path name on drive C: is as follows:

\DBASE\DATA\BUDGET.DBF

The destination full path name for the copied version of the file on drive C: is as follows:

\KEEP\BUDGET89.DBF

To copy the file while logged to the \DBASE\DATA directory, issue the following command:

COPY BUDGET.DBF \KEEP\BUDGET89.*

Because C: is the logged drive, and \DBASE\DATA is the current directory, only the file name and extension are offered as the source parameter. The drive specifier is omitted from the destination parameter because C: is the logged drive. The path to the \KEEP directory is given as well as the changed root file name. The * wild card is given for the destination extension; the command defaults to the .DBF extension of the source parameter when copied.

The same COPY operation is performed while logged to the destination directory of \KEEP using the following form:

COPY \DBASE\DATA\BUDGET.DBF BUDGET89.*

This command gives the full path name for the source file in the \DBASE\DATA directory so that DOS can find the file. The destination parameter is simply the changed root file name and the * wild-card extension. Because \KEEP is the current directory of the logged drive C:, the drive and path parameters are omitted from the destination parameter.

The same COPY operation can be performed from another directory other than the two directories involved in the command's source and destination. In this case, assume that the logged directory is the root. From the root, you issue the following command:

COPY \DBASE\DATA\BUDGET.DBF \KEEP\BUDGET89.*

In this command, the source and destination drives are omitted because C: is the logged disk drive. The full path specifiers are given for the source and destination files because neither the source nor the destination directory is the current working directory.

To illustrate the use of the .. (dot-dot) shorthand name for a parent directory, the following example copies the same file as in the preceding example to the \DBASE directory from the \DBASE\DATA directory. You log to the \DBASE\DATA directory as the current directory and issue the following command:

COPY BUDGET.DBF ..\BUDGET89.*

Notice that the destination parameter includes the .. specifier. From the point of view of the current directory, the command says, "Copy my BUDGET.DBF

Cue:
When using the COPY command, use the .. specifier to refer to any subdirectory's parent directory.

file to my parent directory (\DBASE) and then change the file's name to BUDGET89.DBF." The .. specifier can be used in a file-level command to refer to any subdirectory's parent directory.

Copying from Disk to Disk

With word processors, spreadsheets, and database managers providing much of the useful work on PCs, moving data from one PC to another is a common activity. Unless you work in a networked PC environment, you most likely move data files to or from another computer by way of diskettes. The transportability of floppies is one of their strong features. Whether the task is transporting data to another computer or copying standby files as a "just-in-case" measure between backups, the task requires disk-to-disk copying.

For the first disk-to-disk COPY example, assume that the hard disk drive C: contains three memos with a date in Julian-calendar style as their common extension. (A day's Julian date is the number of days passed since January 1.) The files are located in the \WP\MEMOS directory. The files are named MEETING.233, ESTIMATE.233 and CHKLIST.233.

Cue:
Use the COPY command to copy files from one hard disk to floppies that can then be copied to another hard disk.

In order to print these three memos on the department's laser printer, you must copy the files to a floppy and take the floppy to another computer. You do this task as part of your daily computer work, and you always add today's Julian date as an extension to your memos for tracking purposes. To copy the files from their directory on drive C:, you first log to their directory by using the following command:

```
CD \WP\MEMOS
```

Then you issue the COPY command as follows:

```
COPY *.223 A:/V
```

COPY uses the default disk and the default directory to find all files matching the .223 extension as the source parameter. COPY finds the three memos of the day in the directory. The source files are copied to the diskette in drive A:. Notice that the drive name is the only destination specifier. The files are copied to the root directory of the diskette and retain their original full names. The /V switch causes read-after-write verification as the file is copied.

To copy a file from one diskette to another on a single floppy system, you issue the COPY command with a destination drive name of B: even though a drive B: isn't physically included in your system. DOS uses the existing drive A: as a logical drive B: for the operation. To copy the REPLY.MEM file from one floppy to another in a single floppy drive system, issue the following command:

```
COPY A:REPLY.MEM B:
```

DOS reads the source file into its file buffer and then asks you to swap disks in the drive by prompting with the following:

```
Insert diskette for drive B:
and press any key when ready
```

First, you remove the source disk from drive A: and insert the destination disk in drive A:. When you press a key, COPY writes the buffer contents to the second disk. Depending on the size of the file you are copying, DOS may prompt you to swap disks more times. When COPY has completed the writing of the destination file, you will see the following message:

```
        1 File(s) copied

Insert diskette for drive A:
and press any key when ready
```

You can either change the disk in drive A: back to the source disk or leave the destination disk in the drive. When you press a key, the DOS prompt will appear.

Concatenating Text Files

Although you can use COPY to concatenate any two or more files, the concatenation operation is most effective when the files are ASCII text files. Concatenating binary files results in a destination file that is not usable in most cases.

Reminder: *Use COPY to concatenate ASCII text files; concatenating binary files may result in an unusable destination file.*

For the concatenation examples, assume that there are three text files in the current directory. All three have .TXT extensions. The first file is named INTRO.TXT, the second BODY.TXT, and the third ENDING.TXT. The contents of each file is as follows:

File	*Contents*
INTRO.TXT	Once upon a time,
BODY.TXT	there was a file.
ENDING.TXT	And the file was copied.

To join the three files into a third file, you separate the source files with the + operator by using the following command:

COPY INTRO.TXT+BODY.TXT+ENDING.TXT ALL.TXT

The resulting file, ALL.TXT, contains the text from the three source files. To verify ALL.TXT, issue the following command:

TYPE ALL.TXT

TYPE sends the contents of ALL.TXT to the screen; you see the following:

```
Once upon a time, there was a file. And the file was
copied.
```

You can concatenate the same three files into ALL.TXT in another way. First you must determine if the files are in the correct order in the directory. Use the DIR command in the following form:

DIR *.TXT

DOS reports as follows:

```
 Volume in drive C is LAP TOP
 Directory of  C:\WS

INTRO    TXT        17   8-31-89   3:36p
BODY     TXT        18   8-31-89   3:37p
ENDING   TXT        27   8-31-89   3:37p
        3 File(s)   1648640 bytes free
```

The files are in order, and are the only files in the directory that contain the .TXT extension. To concatenate the files without using the + operator, you use a wild card in the source-file name that will match only the three desired text files. You also give the destination file a literal name. The command takes the following form:

COPY *.TXT ALL.ALL

COPY works down the source directory and finds the three text files with the .TXT extensions. The screen report is as follows:

```
INTRO.TXT
BODY.TXT
ENDING.TXT
        1 File(s) copied
```

To confirm that the wild-card form of concatenation has joined the three source files, use the TYPE command as follows:

TYPE ALL.ALL

Again, the screen shows the following:

```
Once upon a time, there was a file. And the file was
copied.
```

If you were to name the destination file in the preceding example ALL.TXT instead of ALL.ALL, the source wild-card parameter would find the destination

file's directory entry at the end of the directory while the operation was in progress. Your screen would then show the following:

```
INTRO.TXT
BODY.TXT
ENDING.TXT
ALL.TXT
Content of destination lost before copy
Content of destination lost before copy
        1 File(s) copied
```

Because COPY created the ALL.TXT entry in the directory to accommodate the destination file name, the destination file name was also in the directory for the source wild card to match. In carrying out the command as specified, DOS had no choice but to truncate ALL.TXT after reading its contents. Fortunately, ALL.TXT was completely read into the buffer before COPY had to truncate the source version of ALL.TXT to create the destination version of ALL.TXT. The final content of ALL.TXT is the same as ALL.ALL. If ALL.TXT had been too large to fit in the buffer at once, the unread portion of the source version of ALL.TXT would have been lost. The unread portion would never have been included in the destination version of ALL.TXT.

For added insurance against destroying part of the destination file of a concatenation, use the + operator to explicitly list the files rather than using wild cards, or give the destination file a unique full name.

Reminder:
Use the + operator to explicitly list the files during a concatenation operation to guard against destroying part of the destination file.

Copying from a Device to a File

A very common and handy use of the COPY command is copying keystrokes entered from the CON device to a file. The resulting text file can be a batch file, a configuration file, a memo, or just about any kind of short, ASCII file.

The use of COPY with the CON device as a source parameter is so popular that many users refer to the "COPY CON command" when talking about the operation. For an example of copying the console device to a file, you can create a simple batch file that changes the current directory to \123R3 and starts Lotus 1-2-3 Release 3. The batch file is copied to the \DOS directory, so it is accessible from any directory (if you have included \DOS as a search-path alternative [see Chapter 6]). The destination file's name is RUN123.BAT. After the COPY operation, LOTUS can started from any disk or directory. The command to create the batch file is as follows:

COPY CON C:\DOS\RUN123

Cue:
Use the COPY CON technique to copy the contents of the keyboard device to a file.

When you press Enter, DOS will display the word CON on the next line of the screen. This is DOS's indicator that CON is the source device and that DOS is awaiting input from the keyboard.

When you see the word CON, enter the following line; be sure to press Enter:

C:

The cursor drops to the next line, and you enter the following line. Again, remember to press Enter.

CD\123R3

The cursor drops to the next line; enter the following:

LOTUS

The cursor drops to the next line, and you press the F6 function key or the Ctrl-Z key combination. Now press Enter one last time. DOS responds with the following:

```
1 File(s) copied
```

To confirm that the new file is correct, you can use the TYPE command to review the contents of the file. If you try this example on your system, be sure to use the appropriate directory names for DOS and Lotus 1-2-3. Also make sure that the start command for Lotus 123 is LOTUS on your system.

Concatenating a File and a Device to a File

You can add additional text to an existing file using the COPY command to concatenate the file's content with keyboard input from the CON device. You may want to add a line to your AUTOEXEC.BAT file or your CONFIG.SYS file. Instead of using a text editor such as EDLIN to edit the file to which you want a line added, you can use COPY.

For this example, assume that your AUTOEXEC.BAT file contains the following:

```
PATH=C:\DOS;C:\DOS\DRIVERS;C:\
PROMPT $P$G
CLS
```

You decide that seeing a wide directory listing of the files in the root directory would be handy as part of your boot process. To add the wide directory command line to your existing AUTOEXEC.BAT file, you issue the following command:

COPY \AUTOEXEC.BAT+CON

DOS displays the following:

```
C:\AUTOEXEC.BAT
CON
```

As you recall from the previous example, when DOS displays the word CON, DOS is waiting for your input. In this case, the contents of AUTOEXEC.BAT is already in DOS's file buffer; what you type will be joined with the existing text.

To ensure that a new line separates the current contents of AUTOEXEC.BAT and the command you will add, press the Enter key. The cursor will drop one line. Now type the following, but do *not* press Enter:

DIR/W

Press F6 or the Ctrl-Z key combination and then press Enter. DOS will respond as follows:

```
1 File(s) copied
```

The old contents of AUTOEXEC.BAT and the keystrokes from CON are concatenated to form the new contents of AUTOEXEC.BAT. Again, to verify the new contents, use the TYPE command. The new contents are displayed as follows:

```
PATH=C:\DOS;C:\DOS\DRIVERS;C:\
PROMPT $P$G
CLS
DIR/W
```

Joining Files in Binary Mode

COPY uses ASCII mode as a default concatenation mode. You can, however, force the binary mode by using the /B switch. To illustrate the difference between binary mode and ASCII mode in concatenation, assume that two text files have been prepared in the current directory. Both files contain the numbers 1 through 9 followed by 0. One of the text files was copied in the ASCII mode and contains a Ctrl-Z character. The DIR command shows the size of the two files as follows:

```
Volume in drive C is LAP TOP
Directory of  C:\WS

PART1    TXT        11   9-01-89  10:34p
PART2    TXT        10   9-01-89  10:36p
        2 File(s)   1683456 bytes free
```

You can see that the PART1.TXT file is one byte longer than PART.2 because of the Ctrl-Z marker in PART1.TXT. Using the TYPE command, you can see the contents of the two files and notice that the Ctrl-Z does not appear as part of TYPE's output.

```
1234567890

1234567890
```

To concatenate the PART1.TXT file, including the Ctrl-Z, and the PART2.TXT file, issue the following command:

COPY PART1.TXT/B+PART2.TXT TOTAL.TXT

The concatenation operation results in the TOTAL.TXT file which contains 21 bytes. In the file, the first 10 bytes are the number sequence. The next byte is a Ctrl-Z. The remaining 10 bytes are the number sequence again. If you use the TYPE command to see the contents of TOTAL.TXT, you will only see the first sequence of numbers. TYPE encounters the Ctrl-Z character after the first number sequence in TOTAL.TXT and stops outputting the file. To TYPE, the Ctrl-Z is the end-of-file marker even though 10 more bytes are in the file.

You can see the full content of TOTAL.TXT by using the COPY command in the binary mode. In this case, you copy from the TOTAL.TXT file to the CON device. You issue the following command:

COPY TOTAL.TXT/B CON

The screen displays as follows:

```
                          CON
1234567890 -->-->--> 1234567890         1 File(s) copied.
```

The binary copying of TOTAL.TXT to the console sends the Ctrl-Z character to the screen and does not interpret the Ctrl-Z as the end of file. The series of right arrow characters is DOS's display representation for Ctrl-Z. You can see that the Ctrl-Z character (the right arrow sequence) is positioned between the two sets of numbers. DOS's messages do not appear on new lines in this example because TOTAL.TXT does not contain any new line characters produced by the Enter key.

Turning Verification On with VERIFY

Cue:
The DOS VERIFY command controls DOS's internal verification system.

The VERIFY command controls DOS's internal verify indicator. The verify indicator controls the read-after-write checking of data written to disk. If the verify indicator is on, data written to a disk is verified. If the verify indicator is off, the data written to disk is not verified.

DOS does not perform the actual verification but simply maintains the setting of the verify indicator. Data verification is the job of the device driver. The exact method a device driver uses to verify data written to a disk varies. Thorough device drivers may read a written sector to compare with the contents of the disk buffer in memory. The device drivers supplied as a part of the BIOS extension layer of DOS check the written data by rereading the data to verify that the data can be read and that the data's CRC is correct. The DOS BIOS device drivers do not compare the reread data with the contents of the buffer.

You can verify operations with the COPY command by including the /V (verification) switch in the command. This switch turns on the verify indicator for the duration of the copy operation. XCOPY also has a /V switch. Other commands, however, must rely on the action of the DOS VERIFY command to take advantage of the read-after-write verification of data written to disk. Commands such as BACKUP, RESTORE, XCOPY, DISKCOPY, and REPLACE benefit from the VERIFY command.

Reminder:
You can verify operations with the COPY and XCOPY commands by using the /V switch with those commands.

Verifying data during disk writes is inherently slower than not verifying data. Each disk write is reread when VERIFY is on. The reread sector's CRC (Cyclic Redundancy Check) is checked for correctness. The CRC is a reliable indicator of write errors. Even the fact that the written sector can be found is a verification of correct operation. If the sector cannot be found or if the CRC is erroneous, DOS assumes that the data is incorrectly written. This extra verification activity takes extra time. In most disk-intensive activities, a program will be 20% to 40% slower during disk I/O. If you want the extra insurance of data verification, however, the slower disk operation is a small price to pay.

Reminder:
Even though data verification is time-consuming, it provides extra insurance for your files.

Issuing the VERIFY Command

The VERIFY command can be issued in any of the following three forms:

VERIFY ON

VERIFY OFF

VERIFY

When you boot the computer, the default setting of VERIFY is off. Use the first form of the VERIFY syntax to turn VERIFY on.

When VERIFY is on, use the second form of VERIFY syntax to turn VERIFY off.

To see the current setting of VERIFY on-screen, use the VERIFY command with no parameter.

Understanding the General Rules of VERIFY

- If you issue VERIFY with a parameter other than ON or OFF, DOS issues the message `Must specify ON or OFF`.
- If you issue VERIFY with no parameter, DOS will respond with `VERIFY is ON` or `VERIFY is OFF`.
- If VERIFY is on, specifying a /V switch in a COPY command will have no additional verification effect.
- You cannot use VERIFY to cause verification of a network drive.

Verifying a Critical BACKUP Operation with VERIFY

If you plan to perform a critical backup operation on your hard disk, you can add some extra insurance to the operation by turning the verify indicator on. DOS will verify each sector as it is written to the backup disk(s). To turn VERIFY on, issue the command **VERIFY ON**.

You can confirm the setting at any time from the DOS prompt by issuing the command **VERIFY**. DOS will respond as follows:

```
VERIFY is ON
```

When you no longer want verification, you can turn the feature off with the command **VERIFY OFF**.

Caution: *Don't confuse the VERIFY command with a SET command variable.*

It is easy to confuse the VERIFY command with a SET command variable. If you issue the command **SET VERIFY OFF**, DOS will issue the following error message:

```
syntax error
```

No syntax error message is reported, however, if you issue the command **SET VERIFY=OFF**.

Although you have given DOS a command that was accepted, you have not affected the current setting of VERIFY. Instead, you have assigned the value of OFF to an environment variable named VERIFY. You can see the environment variable with the other variables by issuing a SET command. (The SET command is covered in Chapter 8.) DOS will display the following:

```
COMSPEC=C:\COMMAND.COM
PATH=C:\DOS;C:\DOS\DRIVERS;C:\
PROMPT=$P$G
VERIFY=OFF
```

The last variable displayed seems to indicate that VERIFY is OFF. The real meaning of the line is that the environment variable called VERIFY has a string value of OFF. This environment variable is not related in any way to the internal verify indicator controlled by the VERIFY command. To prove that VERIFY is still on, issue the VERIFY command with no parameters; DOS will report the following message:

```
VERIFY is ON
```

Unlike PROMPT and PATH, VERIFY does not export its current setting to the environment. When you issue a PROMPT or PATH command, the command's parameter is placed in the environment. DOS uses the environment string to get the setting of the prompt or the search path. VERIFY is an internal indicator

or flag. DOS does not look in the environment for a VERIFY variable. DOS (or the device driver) checks the internal flag.

Making Backup Sets with BACKUP

Compared to floppy disks, hard disks have many advantages. Hard disks are faster, have greater storage capacities and larger root directories, support multiple partitions, and never require a disk swap. A file on a hard disk can be many times the size of a file on a floppy disk. DOS provides the external BACKUP command and its counterpart, the RESTORE command (discussed in the last section of this chapter) to enable users to maintain diskette backup sets of a hard disk's contents.

BACKUP is a modified COPY command that copies files from one disk, normally a hard disk, to another disk, normally a floppy disk. The "copies" of the files on the second disk are held in a modified format which cannot be accessed by DOS commands, other than the RESTORE command. BACKUP does not use the normal file format on the backup disk. Only the BACKUP and RESTORE commands can use this special file format.

Reminder: *BACKUP and RESTORE use a special format when storing files.*

Backup disks not only contain special copies of files; they also contain directory information about each file that enables RESTORE to copy the backup files to their original locations in the directory tree. In a way, the backup disks are a picture of the original files and directories held in suspended animation.

BACKUP can spread a single file across more than one diskette. No other DOS command can cross a disk's FAT boundary with a single file. BACKUP's capability to cross disk boundaries with a single file allows BACKUP to "copy" to additional diskettes files that are too big to fit on a single diskette. If a BACKUP operation uses more than one diskette, each diskette is internally linked to the next diskette to form a backup set.

Backup set disks are related by the order in which they were created by BACKUP. Each disk in a set is filled to capacity until the files from the disk being backed up are complete. If a file is only partially written to a backup disk when the disk reaches full capacity, the remainder of the file is written to the next disk in the set.

Cue: *You can specify a directory or branch of directories, a file name, a wild-card pattern, and switches with the BACKUP command.*

BACKUP is a selective command. You can specify a directory or branch of directories, a file name, a file name patterned after a wild card, and additional selection switches. By taking advantage of BACKUP's selectivity, you can maintain more than one set of backup disks at one time, with each set having a logical purpose.

You can, for example, keep one backup set that contains every file on your hard disk. This set is insurance against data loss resulting from a hard disk failure or crash. You may have another backup set that contains only the files which have

their archive attributes turned on. With an archive set for each day of the week, you can recover a week's worth of data between complete backups. When your hard disk fails, you replace it, restore using the full set, and then update with the daily set. Whatever the logical method for using BACKUP, the very presence of the BACKUP and RESTORE commands indicates that a reason exists to replicate your data.

Following are some of the reasons that you should always use some sort of backup system and replicate your data:

- Disk media failures
- Disk hardware failures
- Disk controller failures
- RAM or ROM failures
- Program bugs
- Human Error
- So-called "computer viruses"
- Deliberate sabotage

Any of these reasons can destroy data on your hard disk. If you have not replicated the data, you may be forced to recreate the data wherever possible and suffer the consequences of permanent data loss where you cannot recreate data. If you have never experienced a disk-related failure, don't be overconfident. With 10 million hard-disk based DOS users working with their computers for an average of eight hours per day, over 250,000 people will experience a hard-disk related failure this year. You could be next.

Reminder: *Every PC user should develop a backup policy.*

Every PC user should develop a backup policy. A backup policy is simply a defined method of backing up data on a regular basis. Your policy may include making and keeping more than one backup set. Before you decide how to implement a backup policy, you should be familiar with the BACKUP command and its switches. You can pick up on the topic of the backup policy after the following section on BACKUP's syntax.

Issuing the BACKUP Command

The syntax for the BACKUP command is as follows:

BACKUP *d1:path\filename.ext* **d2:** */switches*

d1: is the optional name of the source drive that you are backing up. If you omit the drive source name, the default drive is assumed.

path\ is the name of the optional path holding the files you are backing up. If you omit the path name, the current directory of the specified (or default) drive is assumed.

filename.ext is the optional name of the source file(s) that you are backing up. You can use wild cards in the root file name as well as in the extension. If you omit the file-name specifier, the *.* wild-card pattern is assumed.

d2: is the required name of the destination drive where the backup disks will be written. If the destination drive name is omitted, DOS will issue an error message.

/*switches* are optional switches that add additional selectivity and control to the command. Table 7.4 outlines each switch and the DOS versions with which the switch can be used.

Table 7.4
BACKUP Switches

Switch	*DOS Version*	*Action*
/S	2.0 or later	Includes files from the specified directory and all subsequent *subdirectories* of the specified directory.
/M	2.0 or later	Includes files whose archive attribute is turned on.
/A	2.0 or later	*Adds* the files to be backed up to the files existing on the backup set.
/F	3.3 or later	*Formats* destination disks as part of the backup operation.
/F:size	MS-DOS 4.0	Optional *format size* is given as /F:size.
/D	2.0 or later	Includes files created or modified on or after the *date* specified with the /D switch in the form /D:mm-dd-yy or /D:mm/dd/yy.
/T	3.3 or later	Includes files created or modified on or after the *time* specified as /T:hh:mm:ss.
/L	3.3 or later	Writes a backup *log* entry in the file specified as /L:d:path\filename.ext or in a file named \BACKUP.LOG in the source disk when /L with no file specifier is used.

Understanding the General Rules of BACKUP

- Files backed up are selected by the drive, path, and file specifiers or their default values. Further selection of files is based on the optional switches.
- Unless the destination disk is a backup disk and you use the /A switch, the contents of the destination disk's root directory will be lost in the backup operation.
- If the destination disk is a hard disk, BACKUP checks for a directory named \BACKUP to store the backed up files in. The BACKUP command creates the \BACKUP directory if it does not exist already.
- If \BACKUP contains previously backed-up files, they are erased as part of the backup operation unless the /A switch is used. If the /A switch is used, and the destination disk is a hard disk, the current backup files are appended to existing backup files in \BACKUP.
- DOS V4 automatically formats the destination diskette if it is not already formatted as though the /F switch had been used.
- If BACKUP is formatting destination disks, the FORMAT command must be available in the current directory or accessible through a search path specified by the PATH command. The specified search path in the PATH command cannot be the destination drive.
- BACKUP will not format a hard disk even if the /F switch is given.
- The source disk cannot be write-protected because BACKUP resets each backed-up file's archive attribute.
- BACKUP does not back up the hidden DOS system files or COMMAND.COM in the root directory.
- You cannot use RESTORE V3.2 or earlier versions to restore files backed up by BACKUP V3.3 or later versions.

Determining a BACKUP Policy

Caution: *Uninstall all copy-protected software before you perform a full backup. Copy-protected software will not restore properly.*

The cost of performing a backup is time. The risk of not performing a backup is data loss. The gain of doing a backup is having recoverable data when original data is lost. You must weigh the cost, risk, and gains involved with backing up to determine how often is often enough to back up. The cornerstone of a good backup policy is the periodic full backup.

When you perform a full backup, all of your files are backed up. If disk disaster should strike, you can recover your data from your full backup set. You can restore the full backup set to your existing hard disk or even to a new replacement disk.

Unless you have DOS V3.3 or V4, you need to format the destination disk for the full backup set. The number of disks you need is derived from the number of bytes you will back up. When you run CHKDSK, DOS reports the number of bytes in various categories of files. A sample CHKDSK report looks like the following:

```
Volume LAP TOP      created Aug 17, 1989 6:41p

 21204992 bytes total disk space
    51200 bytes in 3 hidden files
    98304 bytes in 42 directories
 19404800 bytes in 1150 user files
 1650688 bytes available on disk

   655360 bytes total memory
   598784 bytes free
```

Two lines in this report apply to a disk-count calculation for BACKUP: the hidden files line and the user files line. If the number of hidden files is 2 or 3 and the space that the hidden files occupy is less than 150,000 bytes, you can ignore the line. Otherwise, add the number of bytes in hidden files to the number of bytes in user files. The result is the approximate number of bytes in files that you will be backing up. In the preceding case, the number of bytes is rounded to the nearest 10,000 to 19,400,000 bytes.

Divide the rounded number by the capacity of your floppy disk drive. You can multiply your drive's capacity in K by 1,000 or in M by 1,000,000 for a conservative estimate. In this case, you will need about 54 360K diskettes, 27 720K diskettes, 16 1.2M diskettes, or 14 1.44M diskettes. You can also use the information in table 7.5 to derive an approximation of your diskette requirements.

Table 7.5
Number of Diskettes Required for a Full Backup

Bytes To Back Up	*Diskette Capacity*			
	360K	720K	1.2M	1.44M
10M	29	15	9	8
20M	59	29	18	15
30M	83	44	27	22
40M	116	58	35	29
70M	200	100	60	50

Issuing the Full BACKUP Command

Unless you use V4 or intend to use the /F switch with V3.3, format all of your backup floppies the first time you intend to use them. Label each diskette as

"Full Backup Set." When you have formatted and labeled all diskettes, you can start the full BACKUP with the following command:

BACKUP C:\ A:/S

This command tells BACKUP to start in the root directory of the hard disk and process all subdirectories on the disk. BACKUP will issue a warning message.

```
Insert backup diskette 1 in drive A:
Warning! Files in the target drive A:\ root directory
will be erased.
Strike any key to continue.
```

Write the words "disk 1" on the first diskette's label as indicated in the screen message. Then insert the disk and press a key. DOS reports file names on-screen as they are being backed up. When the first disk is filled, BACKUP will repeat the message asking for the next disk. Again, mark the next disk's label with the sequential number that appears in the message.

The process repeats until the last file from the hard disk is processed by BACKUP. When you have completed the full backup, store the destination disks in order and in a safe place. Be sure not to mix this set with another backup set. You might accidentally try to restore a disk from the wrong set later.

Following Up with Incremental Backups

Cue:
The key to making an incremental backup is the archive bit.

An incremental backup is a method of backing up only the files that are no longer current on the full backup set. You use a second set of disks for the incremental backup set. The number of disks in the incremental backup set depends on the number of bytes in files that have been modified since the last BACKUP. The key to making an incremental backup is the archive bit.

As you recall, BACKUP turns off the archive attribute for any file that is backed up. When you modify or create a file, DOS turns on the archive attribute for the modified or created file. BACKUP uses the /M switch to select files with their archive attribute in the on (set) state. Unless you have issued the ATTRIB command with the +A parameter or the XCOPY command with the /M switch, only changed or modified files will have their archive attributes set.

There are two basic strategies for making incremental backups. Either strategy can be combined with the full backup to comprise your backup strategy. Either incremental backup strategy uses the /M switch available with all versions of BACKUP.

The first incremental strategy or method is maintaining several small backup sets and using different sets over time. The second incremental backup method is maintaining one incremental backup set and adding files to the set over time. If your version of BACKUP supports the /A switch, the additive method is for you.

You can perform a full backup weekly or monthly or at some other interval. The interval between incremental backups should cover your risk of data loss between full backups. The time between performing incremental backups using either method depends on the amount of risk you are willing to take. In a business setting, you might reduce your risk to an acceptable level by performing an incremental backup every other day. If your PC activity level is high, you might need to perform an incremental backup daily. If your PC activity is minimal, once every two weeks might be often enough to reduce your risk to a manageable level.

If you use the multiset incremental backup method, you need to ensure that you have on hand enough formatted backup disks to complete any given incremental backup. If you always keep on hand a few spare formatted backup disks, you won't be caught short when BACKUP prompts you to insert another disk.

If you use PC DOS V4, you need only keep on hand blank disks; PC DOS V4 BACKUP detects blanks and formats them on the fly. With MS-DOS V3.3 and later versions, you can always specify the /F switch and format every disk during the backup operation. Using the /F switch will slow the BACKUP operation, but at least you won't be caught with only blank disks and no way to complete the backup.

If you perform an incremental backup every Tuesday and Friday, and you run a full backup every two weeks, you will have four incremental backup sets accumulated before you need to repeat the backup cycle and reuse the disks. If you perform an incremental backup every Friday and a full backup on the last Friday of every month, you will need four sets of incremental backup disks, although most months would only require three. Note that the number of disks in each set will vary with the incremental time period's file-modification activity. The second backup set, for example, may be two disks one month and three disks the next month. You may need to renumber some disks from one set to another to accommodate the peaks and valleys of each set's disk requirements.

If you use the additive incremental backup method, you will have just one incremental backup set. You add files to the end disk in this set when you perform the incremental backups. An exception to adding files to the set occurs during the first incremental backup operation after a full backup. For this first incremental backup in a backup cycle, you start with the first disk in the set by omitting the /A switch from the BACKUP command line. By using this switch, you effectively reclaim the disks for reuse. The number of disks your set requires will depend on the worst case file-modification load for your full backup cycle. If you have a full backup cycle of one month, you may need five disks in your additive incremental backup cycle for 11 months out of the year, but you may need as many as ten disks during a month when you have a lot of file-creation or modification activity. Remember that you can scan the archive attributes of the files in your most heavily-used directories to get a sense of the number of bytes awaiting backup.

Issuing the Multiset Incremental Backup Command

You start one of a multiset incremental backup with the following command:

BACKUP C:\ A: /S/M

The /S switch includes all directories on the hard disk in the operation while the /M switch selects only files with their archive attribute switch set. The backup will proceed as the full backup described previously. Mark each disk in the set with the number and the date belonging to the increment for this backup. Store the completed set in a safe place and avoid mixing the disks with disks from another backup set.

If you use the additive incremental method, you will use the same command as the multiset backup method just described for the first incremental backup following a full backup. For the first incremental backup of the cycle, you will insert disk 1 of the set first.

For every other additive incremental backup besides the first one of the cycle, issue the BACKUP command as follows:

BACKUP C:\ A: /S/M/A

BACKUP sees the /A switch and knows to add the files to the end of the backup set. BACKUP then issues the following prompt:

```
Insert last backup diskette in drive A:
Strike any key when ready
```

You now insert the final disk from the last incremental backup operation. Don't start with the next reclaimed (unused in this cycle) disk. **BACKUP /A** needs to start off exactly where it left off in the last backup operation. When you have inserted the correct disk, press a key. The operation will proceed in the same manner as the full backup described previously.

Restoring Backed-up Files with RESTORE

Cue:
RESTORE reads the special file format copied with the BACKUP command and restores the files to regular format.

The external command RESTORE is the functional counterpart of the BACKUP command. RESTORE is the only command that can read the special control information associated with files that have been copied in the BACKUP format. When you need to recover backup files from a backup set, RESTORE is the command you will use.

By issuing a literal file-name parameter with RESTORE, you can restore a single file from a backup disk. With a directory parameter and a wild-card file-name parameter, you can restore selected files to the named directory. With the /S

switch, you can restore an entire hard disk with only a source drive specifier. RESTORE puts the selection of files in your control.

Like BACKUP, the RESTORE command provides additional control and selectivity through the use of numerous switches. With RESTORE's switches, you can use time, date, presence, and absence as selection criteria for files to be restored.

Issuing the RESTORE Command

The syntax of the RESTORE command is as follows:

RESTORE ds: *dd:path\filename.ext /switches*

ds: is the required source drive that holds the file(s) to be restored.

dd: is the drive that will receive the restored files.

path is the optional path to the directory that will receive the restored files. If the path specifier is omitted, the current directory of the destination disk is assumed.

filename.ext is the optional name of the file(s) to be restored. You can use wild cards in file name's root and extension.

/switches are optional switches.

Table 7.6 displays the optional RESTORE switches, their actions, and the applicable versions of DOS.

Table 7.6
RESTORE Switches

Switch	*DOS Version*	*Action*
/S	2.0 or later	Restores files to the specified (or default) directory and *subdirectories.*
/P	2.0 or later	*Prompts* to confirm the restoration of a read-only file or a file modified since the last backup.
/M	3.3 or later	Restores files *modified* or deleted since the last backup.
/N	3.3 or later	Restores files that *no* longer exist on the destination disk.
/B	3.3 or later	Restores files created or modified *before* or or on the date given as /B:mm/dd/yy or /B:mm-dd-yy.

Table 7.6—*continued*

Switch	*DOS Version*	*Action*
/A	3.3 or later	Restores files created or modified *after* or on the date given as /A:mm/dd/yy or as /A:mm-dd-yy.
/L	3.3 or later	Restores files created or modified *later* than or at the time given as /L:hh:mm:ss
/E	3.3 or later	Restores files created or modified *earlier* than or at the time given as /E:hh:mm:ss.

Cue:
Time and date entries used with RESTORE are given in the format of your country code (see Chapter 15).

Note that time and date entries used with applicable systems are in the format of your country code. United States formats are presented in table 7.6. More information about providing country information can be found in Chapter 15.

Understanding the Operation of RESTORE

RESTORE is the only command with the capability to read the special file format of files "copied" by the BACKUP command. RESTORE always restores files to the directory in which they were located when the files were backed up. RESTORE cannot restore files to another directory other than the files' original directory.

If RESTORE is restoring a file whose original directory is no longer on the destination disk, RESTORE will create the directory before restoring the file. RESTORE makes directory entries for files that are no longer on the destination disk and allocates the next available space in the FAT table for the restored file's allocation. If the destination disk is newly formatted, the restored files on the disk will be contiguous in most cases. Files with the same full path name, but different sizes, from an additive backup set may result in a fragmented destination file after RESTORE allocates the first file of a duplicate name pair and subsequently allocates additional clusters for a larger file with the same name.

Reminder:
RESTORE always restores files to the directory in which they were located when the files were backed up.

Like the BACKUP command, RESTORE prompts you for the disks of the backup set. If you know the disk number which contains the file(s) that you want to restore, RESTORE will accept that disk as the first disk. You can use the /L switch with BACKUP to create a *log* file showing the names and disk numbers of files in a backup set.

Understanding the General Rules of RESTORE

- RESTORE V3.2 or earlier versions cannot restore files created by BACKUP V3.3 or later versions.
- You must give a source drive name in the command line.
- All files restored from the backup set will be placed in their original directories on the destination disk. If a source directory does not exist on the destination disk, the directory will be created.
- Files selected for restoration are selected from the destination drive, path, and file specifiers. Further selection is based on the included switches.
- The destination disk must already be formatted. RESTORE, unlike BACKUP, has no provision to format the destination disk.
- Files restored to a freshly formatted disk will not be fragmented.

NOTE: Copy protected files will not properly restore to the destination disk. Uninstall all copy protected files prior to BACKUP and reinstall the files after RESTORE.

Using RESTORE Following a Disk Failure

For the examples of the RESTORE command, assume that you are using a backup policy that includes a weekly full backup on Friday and an incremental backup each Wednesday. You will have two backup sets. The first set from Friday contains all files. The second set contains only files that were modified or created after Friday, but before Thursday.

On Thursday morning, suppose that you are saving a worksheet file when a workman begins to use a large drill next door. You notice that DOS reports the following message:

```
General Failure on drive C:
```

You abort the spreadsheet session and run CHKDSK to ensure that your FAT and directory system are in order. (The operation of CHKDSK is covered in Chapter 5.) DOS reports hundreds of lost clusters, and CHKDSK fills your root directory with reclaimed clusters. CHKDSK aborts after filling the root directory of your hard disk.

You have had an electrical noise-induced hard disk failure. Your hard disk most likely wrote errant information over valid information in the FAT. You have no choice other than to reformat your hard disk and then fall back to your backup disks. After you format your hard disk, copy the external DOS commands from

the DOS master disk or its working copy back to the hard disk. Use the PROMPT command to set a search path to the DOS directory so that DOS can locate the RESTORE command.

In the case of total disk failure, you restore your backup sets in chronological order. The first set to restore after you format your hard disk is your full backup set. Locate the disks from your full backup set and put them in their proper order. Then issue the following command:

RESTORE A: C:\ /S

DOS will prompt as follows:

```
Insert backup diskette 1 in drive A:
Strike any key when ready.
```

Put the first disk in drive A: and press a key. DOS will respond:

```
*** Files were backed up mm/dd/yy ***
```

The actual message will show mm/dd/yy as the date of the backup and will be in the format used by your country code. DOS then prompts as follows:

```
*** Restoring files from drive A: ***
```

You will see the full path names of the files as they are being restored. When all the files from the first disk are restored to the hard disk, DOS will prompt you for the next disk in the backup set. Repeat the restoration disk prompt cycle until you have completed the restore operation.

With the full backup set restoration complete, you now must restore the incremental backup set. Because you want to restore all the files in the incremental backup set, you use the same command as you used to restore the full backup set. The operation will proceed in the same fashion.

With both backup sets restored, you should run CHKDSK to ensure that the hard disk is in order. Keep both of your backup sets intact until you have determined that your hard disk is performing correctly. Run CHKDSK several times during the day. If all is in order, perform a full backup at the end of the day.

Using RESTORE Following an ERASE *.* Command

In another example of the use of RESTORE, assume that you have accidentally issued an ERASE command with a *.* wild card parameter. You thought you were logged to your temporary directory \TEMP, but instead you were logged to your \REFLEX\DATA directory. Now all of your Reflex data files are erased. To recover your lost data, you will have to perform a selective restoration oper-

ation from your two backup sets. To restore only the files in the \REFLEX\DATA directory from the full backup set, issue the following command:

RESTORE A: C:\REFLEX\DATA*.*

As in the complete restoration example, RESTORE will prompt you to insert the first disk. You can insert the first disk or, if you know the starting disk number that holds the Reflex files, you can insert the disk that holds the Reflex files. RESTORE will bypass any files on the source disk that are not included in the destination parameter that you gave in the command. When RESTORE does encounter a \REFLEX\DATA file, that file will be restored to the \REFLEX\DATA directory of the hard disk.

When you are sure that all files have been restored to the erased directory, you can press Ctrl-C or Ctrl-Break to abort the RESTORE operation. For safety's sake, you should insert all disks from an additive backup; a few files may have been backed up toward the end of the set.

Again, you nust restore any files from the incremental set that belong in the \REFLEX\DATA directory. To restore from the incremental backup set, use the same command that you issued to restore the \REFLEX\DATA files from the full backup set.

If you maintain more than one incremental backup set, you should restore desired files from each of the incremental backup sets you have created since the last full backup and before the incident that caused the need to restore. Always restore backup sets in the order that they were created. Restore the oldest set first and the newest set last. If you restore the sets out of chronological order, you may overwrite more recent versions of files with older versions of the same files.

Caution: Always restore backup sets in chronological order. If you restore the sets out of chronological order, you may overwrite more recent versions of files with older versions of the same files.

In situations which lend themselves to BACKUP and RESTORE, and which do not involve data loss (such as moving data from a work PC to a home PC), you can use the various RESTORE switches to ensure that you aren't moving older data to a PC that contains newer data. You can, for example, restore older files to a PC while preserving files modified or created after a specified date. Use the switches available with RESTORE to get the most from the command.

Summary

This chapter has covered the DOS file-management commands. Files are an essential "way of life" in personal computing. The more competent you are at managing your files, the better your computing experience becomes. In this chapter you learned about the following key file-management points:

- DOS allows a defined set of characters in "legal" file names.

- ❏ DOS devices have names that are reserved. If you use a reserved name for a file specifier, DOS will use the specifier as a device.
- ❏ Some file extensions are conventionally used to indicate the type of file that the extension implies.
- ❏ The ^Z character marks the end of many ASCII files.
- ❏ The ATTRIB command displays and manages a file's archive and read-only attributes.
- ❏ The RENAME command changes the name of an existing file.
- ❏ The ERASE (DEL) command removes files from a directory. Erased files are no longer available to DOS.
- ❏ The COMP and FC commands compare files for differences.
- ❏ The TYPE command displays a file's contents in ASCII characters on-screen.
- ❏ The COPY command concatenates and copies files and devices in binary and ASCII modes.
- ❏ The VERIFY command manages DOS internal verify indicator that device drivers check to determine if the device drivers' need to perform write verification of data.
- ❏ The BACKUP command is used to produce special backup disks. Backup disks are usually used for recovering lost data.
- ❏ The RESTORE command reads files from backup sets and restores the files to their original directories on another disk.

In the next chapter, you will learn the DOS commands that access, display, and change system information. These commands are generic to all phases of PC operation.

8

Working with System Information

Chapters 1 through 4 of this book introduced you to the DOS fundamentals and how they relate to you and the computer hardware. Chapters 5 through 7 showed you the commands associated with the disks, directories, and files. This chapter covers commands that allow you to control how DOS interfaces with you.

These commands can change system information, such as the date and time assigned to files or how the display appears. If you have DOS V4, you might use a command in this chapter to invoke the DOS Shell program. Even if you don't have DOS V4, several commands allow you to adjust how the screen looks. The PROMPT command changes the command prompt displayed for you to enter commands. The change code page command (CHCP) changes the character set displayed, allowing you to use different languages. The MODE command, which has extensive options, can be used to alter the look of the screen by changing the number of characters or lines displayed.

The system information commands covered in this chapter are fully described in the Command Reference. The Command Reference includes all the possible parameters and switches. The commands in this chapter are covered as they relate to system information. For example, the MODE command is covered in this chapter as it relates to altering the display mode, and again in Chapter 9 as it is used to alter input and output devices. The Command Reference provides a complete summary.

The commands covered in this chapter include the following:

- DATE
- TIME
- VER
- MEM

- DIR
- DOSSHELL
- PROMPT
- CHCP
- MODE
- COMMAND
- EXIT
- SET

Changing the Date and Time with DATE and TIME

When DOS saves a new file to disk, the operating system assigns a date and time stamp to that file. This stamp is put into the directory entry along with the file name, attributes, and starting cluster information for the FAT. The date and time assigned to a file are what the computer thinks is the current date and time.

If you have an AT class of machine or a 386-based computer, you have a battery-backed clock that will keep the date and time in an area of memory, once the date and time have been set. Even when the computer is turned off, this small area of memory has power from the battery, and the date and time continue to be current. In a PC or XT, you may have added a battery-backed clock that serves a similar function. The PC-style clocks require you to run a program, supplied with the clock, to transfer to DOS the date and time stored from the memory located on the board containing the clock.

Setting the date and time becomes increasingly significant as you create more files on your computer. DOS has limited file naming capabilities, allowing only eight characters in the root name and three characters in the extension. The convention of using the extension to indicate the file type leaves a limited choice of names for the root name. On computers without a clock, or on computers with a clock and dead batteries, the date and time are set to the same value each time the computer is switched on. Unless you enter the current date and time each time you start the computer, you end up with many files being assigned similar date and time stamps.

Making full use of the date and time facility of DOS allows you to compensate for the file naming limitations of DOS. For example, if you regularly write letters to the utility company, you may want to name the letters something like UTILITY.TXT. Because you can't have duplicate file names, you have to be cre-

ative in your naming, calling them UTILITY1.TXT, UTILITY2.TXT, and so on. Then you could start with UTIL1.TXT and continue. However, you need to remember which is the most recent letter. By checking the date and time, you can easily determine your latest letter by issuing the directory command (DIR).

Caution: *All clocks in personal computers lose time.*

Checking your computer's date and time occasionally is worthwhile for two reasons. First, the battery eventually will fail and need replacing. Remember that the battery operates even when the computer is turned off. That's how it keeps the date and time current. The other reason is that all clocks in personal computers will lose time. How much time they lose depends on the particular programs you run on your computer. For example, one computer might lose about 20 minutes a month; another might lose about 10 minutes a month.

The clock used to keep the date and time in a PC is controlled by an electronic component in the computer called, logically, the *system timer*. This chip can be considered the heartbeat of the computer. Everything the computer does—adding two numbers together, saving a file to disk, or displaying information on the screen—takes a known amount of time. The system timer provides a regular "pulse" that is used to control all the different functions occurring in the computer.

Cue: *If you do not have an AUTOEXEC.BAT file, DOS prompts you for the date and time at boot time.*

One of these functions is to update the clock that DOS uses to tell the time (and consequently the date). However, certain functions that cannot afford to be interrupted while they are working will tell DOS not to interrupt them until they are finished. A typical example is a communications program that is waiting for a character to be sent from another computer. If DOS is busy doing other things, such as updating the clock, the character may be missed. These programs disable the interrupts—in other words, they tell DOS not to do anything else for the periods of time when they need full control. During these times, the clock is not updated. These periods of time are only a few seconds, but the effect is cumulative, and eventually the effect is significant.

If you do not have an AUTOEXEC.BAT file, which runs automatically when you start your computer, DOS prompts you for the date and time at boot time. When this occurs, DOS shows the current settings and allows you to change them. On an AT class or 386-based machine, you probably can accept the current settings because the date and time are loaded from the battery-backed memory.

Cue: *Use DATE and TIME to change the current date and time settings.*

If you have an AUTOEXEC.BAT file that does not have the DATE and TIME commands in it, DOS does not prompt you for the date and time; DOS instead uses its current settings. You can change the current settings at any time by using the DATE and TIME commands.

Issuing the DATE Command

The syntax for the internal DATE command is

DATE *actual_date*

or

DATE

If you type just **DATE** at the DOS prompt, DOS returns the current setting and prompts you for a new date. You can change the date without being prompted by including the new date on the command line.

The *actual_date* parameter specifies the month, day, and year. The exact syntax depends on the country code set in the CONFIG.SYS file. Country code settings are discussed in Chapter 15. The syntax for *actual_date* can be in one of three forms:

mm-dd-yy or *mm-dd-yyyy* for North America
dd-mm-yy or *dd-mm-yyyy* for Europe
yy-mm-dd or *yyyy-mm-dd* for East Asia

mm is the month, *dd* is the day, and *yy* and *yyyy* are the year.

Understanding the Operation of DATE

To enter the date December 23, 1990, on a machine configured for North America, you would type

DATE 12-23-90

The year can be typed in its full form as 1990 if you prefer. If you use two digits, the 19 is assumed. Instead of the hyphen, you can use periods or slashes as separators. Leading digits are not required. For example, if the month is January, you need only to type **1** and not **01** for the month.

Letting DOS prompt you for the date is a better way to use this command, because the current date is displayed in the correct format and you can copy the syntax rather than having to remember it. A typical DOS sequence might be as follows:

```
C:\>DATE
Current date is Mon 09-18-1989
Enter new date (mm-dd-yy): 12-23-1990
```

Pressing Enter when prompted for a new date retains the current setting. If you don't enter the date correctly or you select a date out of the DOS range, the error message `Invalid date` is displayed and DOS displays the `Enter new date` prompt again.

Notice that DOS displays the day of the week along with the other date information. You can use the DATE command to determine the day of the week for a particular date by changing the date, issuing the DATE command to see the new setting, and then changing the date back again.

Cue: *Use the DATE command to determine the day of the week for a particular date.*

DOS V3.3 and V4 actually change the battery-backed clock as well as the system clock setting when you execute the DATE command. Versions of DOS prior to V3.3 do not update the date permanently in the battery-backed clock; those versions keep the date entered only for the current DOS session. When the power to the computer is turned off, the date value is the one stored in the battery-backed clock. To change the clock permanently for earlier version of DOS, use the SETUP program supplied with your computer.

Issuing the TIME Command

The syntax for the internal TIME command is

TIME *actual_time*

or

TIME

As with DATE, if you type only **TIME** and press Enter at the command prompt, you are shown the current setting and are prompted for a new setting. You can avoid the prompt by typing the desired time as a command line parameter.

The syntax for *actual_time* varies with the COUNTRY setting in the file CONFIG.SYS. A 12-hour or 24-hour clock is possible. The syntax for the 24-hour clock is

hrs:mins:secs.1/100secs

hrs is the hour, a number from 0 to 23; *mins* is the minutes, a number from 0 to 59; *secs* is the seconds, a number from 0 to 59; and *1/100secs* is the number of one-hundredth seconds, a number from 0 to 99.

Understanding the Operation of TIME

The DOS V4 12-hour clock uses the same syntax except that *hrs* can be a number from 1 to 12, and an *a* or *p* is added to signify a.m. or p.m., respectively. You can enter a 24-hour time if you want. DOS will display it in the 12-hour format.

Cue: *If you enter a 24-hour time, DOS displays it in the 12-hour format.*

In either form, there are no spaces between parameters. Also, in the case of the 12-hour version, there is no space between the end of the time elements and the *a* or *p*. If you prefer, you can use a period in place of the colon when

separating the hours, minutes, and seconds; or a comma in place of the period when separating the seconds from the hundredths of a second.

Cue: *You don't have to specify all the parts of the time; any missing elements are set to zero.*

You don't have to specify all the parts of the time. Any missing elements are set to zero (using the 24-hour clock notation). To set the time to 8:25 p.m., you type

```
TIME 20:25
```

or

```
TIME 8:25p
```

Either command sets the time exactly to 8:25 p.m. The seconds and hundredths of a second automatically are set to zero.

To set the clock to 12:30 a.m. you can type

```
TIME :30
```

This command is easy to use if you let DOS prompt you for the time, because DOS shows you the syntax. A typical sequence might be as follows:

```
C:\>TIME
Current time is 2:35:07.23p
Enter new time: 2:40p
```

You can use the COUNTRY command to change the format DOS uses to display the time, as described in Chapter 15.

As with the DATE command, DOS V3.3 and V4 actually change the system clock setting when you execute the TIME command, and earlier DOS versions do not update the TIME in the battery-backed clock. You need to use the SETUP program supplied with your computer to change the clock permanently with these earlier versions of DOS.

Displaying the DOS Version with VER

Cue: *Use VER if you have to troubleshoot someone else's computer.*

DOS includes an internal command that displays the DOS version: VER. This seemingly trivial command is invaluable if you have to troubleshoot someone else's computer. Commands such as DISKCOMP, FORMAT, and XCOPY either work differently or are not available with different DOS versions. New users probably won't know what version of DOS they are using, and the VER command eliminates many possibilities for "my computer doesn't do what I tell it to."

Issuing the VER Command

The syntax for the VER command is

VER

Understanding the Operation of VER

After you issue the VER command, DOS displays a message showing the version of DOS that was used to boot the computer. If a computer with a hard disk is booted from a floppy disk instead of the hard disk, you may have access to a different version of DOS than you were expecting. Suppose, for example, that your hard disk is formatted for V4, but you use a V3.3 floppy disk to boot the computer. In this case, the V4 commands that reside on the hard disk are not available. The DOS message `Incorrect DOS version` is displayed when you try to use one of the V4 commands.

Caution:
If a computer with a hard disk is booted from a floppy disk instead of the hard disk, you may have access to a different version of DOS than you were expecting.

The message displayed by DOS when the VER command is used is of the form

```
MS-DOS Version 3.30
```

or

```
MS-DOS Version 4.01
```

Other manufacturers supply different implementations of DOS, and consequently you will see a different message if you are using a different product. For example, COMPAQ Computer Corporation's version of DOS will include COMPAQ's name along with the version number. Because of the differences between these versions and implementations, you may be able to track down problems on a different machine more easily if you first can determine the exact version number and manufacturer.

Checking Machine Configuration with MEM

DOS V4 includes a new external command called MEM, which can be used to provide system information. The command calculates all the used and unused memory on the system and, in its most advanced form, can be used to see what programs and devices are loaded in the system.

Cue:
Use the MEM command to calculate all the used and unused memory on the system and to see what programs and devices are loaded.

Issuing the MEM Command

The syntax for the MEM command is

MEM */PROGRAM /DEBUG*

Cue:
If you are designing a program to run under DOS, you should make exhaustive use of the MEM command.

Both switches are optional. The */PROGRAM* switch gives a list of all programs currently loaded on the computer. The */DEBUG* switch gives a list of all the programs as well as all the device drivers loaded on the computer. This listing includes the name, size, position, and type of each item. Because */DEBUG* is an extension of the */PROGRAM* switch, you would use only one switch on a single command line.

Understanding the Operation of MEM

Generating an extended listing is only of passing interest unless you are a programmer. The listing allows programmers to identify such things as the order that programs are loaded into memory and their actual size. If you are designing a program to run under DOS, you would make exhaustive use of this valuable command.

Cue:
Run MEM on an unfamiliar computer to gain insight into memory availability.

The basic command without any switches is of value to all users. Like the VER command, MEM is a program you run only infrequently on a particular machine. After you know how much memory is on your computer and how the memory is configured, you usually don't need to run MEM again. The command is worth running on an unfamiliar computer because MEM gives insight into memory availability.

On a computer with only 512K of memory and no extended or expanded memory, you may see the following when you type **MEM** at the command prompt:

```
524288 bytes total memory
524288 bytes available
455456 largest executable program size
```

Cue:
The amount of memory available for a program to run is the total amount of memory less any memory that DOS uses.

The amount of memory available for a program to run is the total amount of memory less any memory that DOS uses. Any device drivers or memory resident programs you have loaded on your computer will reduce the available 455,456 bytes.

On a computer with 640K of memory and an installed 2M memory board, you might see the following when you issue the MEM command:

```
655360 bytes total memory
655360 bytes available
569504 largest executable program size
```

```
1703936 bytes total EMS memory
1703936 bytes free EMS memory

524288 bytes total extended memory
524288 bytes available extended memory
```

All the expanded and extended memory currently is available for use by programs. (The difference between these two types of memory is discussed in Chapter 4.) The configuration of the 2M board can be seen from running the MEM command. Notice that 524,288 bytes (512K) have been assigned to extended memory. The rest is expanded memory. However, 1.5M of memory is 1,572,864 bytes. The extra EMS memory that an applications program can use is taken from the lower 640K of memory on the system as allowed by LIM EMS (Lotus-Intel-Microsoft Expanded Memory Specification). MEM shows the maximum potential EMS memory.

Cue: *The extra EMS memory that an application program can use is taken from the lower 640K of memory on the system as allowed by LIM EMS.*

On a different computer, you may see different results. For example, on a 386-based computer, which has a 386 memory manager (386MAX), a variety of device drivers, and memory resident programs, you might see something like the following when you run MEM:

```
655360 bytes total memory
655360 bytes available
566288 largest executable program size

4292608 bytes total EMS memory
3686400 bytes free EMS memory

4194304 bytes total extended memory
69632 bytes available extended memory
```

This display shows that nearly 4M of extended memory is installed on the machine. The 386 memory manager converts most of this memory to EMS memory, leaving 68K of memory as extended memory. Additional EMS memory is found by the MEM command in the base 640K of DOS memory.

Try running the MEM command from the DOS command prompt and then again from within an applications program. In 1-2-3, the command that handles this task is called /System. This command leaves the worksheet in its existing form and displays a DOS command prompt. Run the MEM command from this command prompt, and you will see a dramatically reduced quantity of available memory.

Cue: *Try running the MEM command from the DOS command prompt and then again from within an application program.*

Executing the MEM command with the program switch by typing

MEM /PROGRAM

might give you results similar to the following:

```
Address     Name          Size      Type
000000                    000400    Interrupt Vector
000400                    000100    ROM Communication Area
000500                    000200    DOS Communication Area
000700      IO            0024E0    System Program
002BE0      MSDOS         0088B0    System Program
00B490      IO            004D10    System Data
              386MAX      000040     DEVICE=
              386LOAD     000180     DEVICE=
              386LOAD     000160     DEVICE=
                          0005D0     FILES=
                          000100     FCBS=
                          003410     BUFFERS=
                          0001C0     LASTDRIVE=
                          000CD0     STACKS=
0101B0      MEM           000040    Environment
010200      MSDOS         000070    -- Free --
010280      FASTOPEN      002720    Program
0129B0      COMMAND       001640    Program
014000      COMMAND       000100    Environment
014110      MEM           012F60    Program
027080      MSDOS         078F70    -- Free --

  655360 bytes total memory
  655360 bytes available
  573152 largest executable program size

 4292608 bytes total EMS memory
 3686400 bytes free EMS memory

 4194304 bytes total extended memory
   69632 bytes available extended memory
```

Items of interest in this listing are the 386 memory manager and a 386 device driver loader, the FASTOPEN program, and the command processor COMMAND and its environment.

The first column shows the starting address of each item that MEM found. The addresses are listed in hexadecimal (base 16). The second column shows the name of the program loaded into memory. The third column shows the length of each program in hexadecimal. Therefore, with a given starting address and known length, you can calculate the complete position in memory for a particular program. The final column includes the type of item listed. The types include the system files IO.SYS and MSDOS.SYS, programs, and any data areas the programs may need.

The /DEBUG switch extends this listing further to include all the device drivers that are loaded into memory.

Listing Files with DIR

The DIR command is one of the first commands most DOS users learn. The command quickly gives a list of files, along with the date and time of creation, and the file sizes. If you type only **DIR**, you are not using the command to its full power.

Cue:
Use DIR to display a list of files, along with the date and time of creation, and the file sizes.

Issuing the DIR Command

The syntax of the DIR internal directory command is

DIR *d:path\filename /P /W*

d: is the drive, *path* is the path name, and *filename* is the name of the file to be displayed. The optional switches, */P* and */W*, are used to display a screenful of files at a time and a wide screen format of the directory, respectively.

Cue:
The DIR command's optional switches, /P and /W, display a screenful of files at a time and a wide screen format of the directory, respectively.

If any of the parameters are missing, the current status is assumed. For example, if you omit the path name, DOS assumes the current subdirectory. Omitting the drive causes DOS to assume the current drive.

Understanding the Operation of DIR

Hard disks are now much more prevalent on computers than they were even a couple of years ago. A hard disk holds many times the quantity of files that can be stored on a diskette. You can end up with directory listings that fill the screen many times over unless you instill some organization into the hard disk. Using hierarchical directories alleviates many of these problems, but also can effectively hide files. The DIR command, coupled with a logical hard disk directory organization, will make all your files easily accessible.

Consider first the simplest form of the DIR command. Typing **DIR A:** at the DOS prompt when the DOS V3.3 Operating disk is in drive A will give you the full listing of the directory.

```
 Volume in drive A has no label
 Directory of  A:\

APPEND   EXE      5825 03-17-87  12:00p
ASSIGN   COM      1561 03-17-87  12:00p
ATTRIB   EXE      9529 03-17-87  12:00p
BACKUP   COM     31913 03-18-87  12:00p
```

```
BASIC    COM      1063 03-17-87   12:00p
BASICA   COM     36403 03-17-87   12:00p
CHKDSK   COM      9850 03-18-87   12:00p
COMMAND  COM     25307 03-17-87   12:00p
COMP     COM      4214 03-17-87   12:00p
DEBUG    COM     15897 03-17-87   12:00p
DISKCOMP COM      5879 03-17-87   12:00p
DISKCOPY COM      6295 03-17-87   12:00p
EDLIN    COM      7526 03-17-87   12:00p
FIND     EXE      6434 03-17-87   12:00p
FORMAT   COM     11616 03-18-87   12:00p
GRAFTABL COM      6128 03-17-87   12:00p
GRAPHICS COM      3300 03-17-87   12:00p
JOIN     EXE      8969 03-17-87   12:00p
LABEL    COM      2377 03-17-87   12:00p
MORE     COM       313 03-17-87   12:00p
PRINT    COM      9026 03-17-87   12:00p
RECOVER  COM      4299 03-18-87   12:00p
RESTORE  COM     34643 03-17-87   12:00p
SHARE    EXE      8608 03-17-87   12:00p
SORT     EXE      1977 03-17-87   12:00p
SUBST    EXE      9909 03-17-87   12:00p
TREE     COM      3571 03-17-87   12:00p
XCOPY    EXE     11247 03-17-87   12:00p
BASIC    PIF       369 03-17-87   12:00p
BASICA   PIF       369 03-17-87   12:00p
MORTGAGE BAS      6251 03-17-87   12:00p
       31 File(s)      55296 bytes free
```

Caution:
The size of the file is not necessarily the same as the amount of disk space occupied by the file.

The information displayed is most of the information stored in the directory table. The file attributes are not included, but are used by DOS to determine whether a file should be displayed. Hidden files are not included in this list. The location of the first cluster in the file allocation table (FAT) for the file is also not displayed because the file's starting cluster is not of use to you.

The volume label and the amount of free space on the disk is also shown in the directory listing. If you were to add up the size of each file on the disk and subtract it from the total disk capacity, you would probably not get the same number of free bytes as that shown in the directory listing. However, the directory listing is correct. Remember that hidden files are not shown, and more importantly, the size of the file is not necessarily the same as the amount of disk space occupied by the file.

Cue:
Free clusters are the positions remaining on the disk that are not allocated to another file.

When a file is created, no matter how small it is, it is allocated a whole number of clusters on the disk. The directory listing shows the file length in bytes and calculates the free space based on the number of free clusters on the disk.

These free clusters are the positions remaining on the disk that are not allocated to another file. (See Chapter 2 for more information about the file storage system.)

Displaying a Screenful of Information with DIR

Taking the same directory listing and using the */P* switch (typing **DIR /P**) allows a screenful of information to be displayed at a time. Pressing any key will display the next page of information. This switch works in a similar fashion to the MORE filter discussed in Chapter 9. For the DIR command, however, */P* is convenient because */P* is built into the DIR command and doesn't require MORE to be in the search path. When the DOS V3.3 Operating disk is in drive A, typing **DIR /P** will give you the following listing of the directory:

```
 Volume in drive A has no label
 Directory of  A:\

APPEND   EXE      5825 03-17-87  12:00p
ASSIGN   COM      1561 03-17-87  12:00p
ATTRIB   EXE      9529 03-17-87  12:00p
BACKUP   COM     31913 03-18-87  12:00p
BASIC    COM      1063 03-17-87  12:00p
BASICA   COM     36403 03-17-87  12:00p
CHKDSK   COM      9850 03-18-87  12:00p
COMMAND  COM     25307 03-17-87  12:00p
COMP     COM      4214 03-17-87  12:00p
DEBUG    COM     15897 03-17-87  12:00p
DISKCOMP COM      5879 03-17-87  12:00p
DISKCOPY COM      6295 03-17-87  12:00p
EDLIN    COM      7526 03-17-87  12:00p
FIND     EXE      6434 03-17-87  12:00p
FORMAT   COM     11616 03-18-87  12:00p
GRAFTABL COM      6128 03-17-87  12:00p
GRAPHICS COM      3300 03-17-87  12:00p
JOIN     EXE      8969 03-17-87  12:00p
LABEL    COM      2377 03-17-87  12:00p
MORE     COM       313 03-17-87  12:00p
PRINT    COM      9026 03-17-87  12:00p
RECOVER  COM      4299 03-18-87  12:00p
RESTORE  COM     34643 03-17-87  12:00p
Press any key to continue . . .

SHARE    EXE      8608 03-17-87  12:00p
SORT     EXE      1977 03-17-87  12:00p
```

```
SUBST    EXE      9909 03-17-87   12:00p
TREE     COM      3571 03-17-87   12:00p
XCOPY    EXE     11247 03-17-87   12:00p
BASIC    PIF       369 03-17-87   12:00p
BASICA   PIF       369 03-17-87   12:00p
MORTGAGE BAS      6251 03-17-87   12:00p
       31 File(s)      55296 bytes free
```

If you want to see more of the files at once, use the */W* switch. That switch displays only the file names in the format shown in figure 8.1, which fits many more files on a single screen.

Fig. 8.1.

File names displayed in a wide format with the /W switch.

```
Volume in drive A has no label
Directory of  A:\

APPEND   EXE    ASSIGN   COM    ATTRIB   EXE    BACKUP   COM    BASIC    COM
BASICA   COM    CHKDSK   COM    COMMAND  COM    COMP     COM    DEBUG    COM
DISKCOMP COM    DISKCOPY COM    EDLIN    COM    FIND     EXE    FORMAT   COM
GRAFTABL COM    GRAPHICS COM    JOIN     EXE    LABEL    COM    MORE     COM
PRINT    COM    RECOVER  COM    RESTORE  COM    SHARE    EXE    SORT     EXE
SUBST    EXE    TREE     COM    XCOPY    EXE    BASIC    PIF    BASICA   PIF
MORTGAGE BAS
      31 File(s)     55296 bytes free
```

Caution:
In the wide directory listing, you can't tell which item is a file and which is a directory.

The wide format requires some careful use. With hierarchical directories, your directory listing includes subdirectory names along with files. In the wide listing, you can't tell which item is a file and which is a subdirectory. If you use the DIR command before deleting a file, be careful that you don't attempt to delete a directory thinking it's a file. If you ever see the following prompt, think before saying Yes:

```
All files in directory will be deleted!
Are you sure (Y/N)?
```

The preceding prompt may mean that DOS has found a path name that is the same as your file specifier. On the other hand, the wide listing is useful when you are trying to see the type of files you have in the directory, because the names are grouped so closely together.

Using the DIR Command with Wild Cards

The DIR command becomes more powerful when used with wild cards. Chapter 6 describes the use of .. and .—the parent and current directories. This technique can be used with the DIR command. Typing **DIR ..** lists the contents of the parent directory, while typing **DIR ..\..** lists the contents of the grandparent directory.

Use the DIR command liberally, especially before deleting files using wild cards. If you type **DIR** followed by the file name, the selected list of files is displayed. If this list contains all the files you want to delete and no more files, use the DOS editing commands to alter the **DIR** to **DEL**. You won't accidentally delete files this way. The following interchange illustrates the technique:

Cue:
Use the DIR command liberally, especially before deleting files using wild cards.

```
C:\MEMOS>DIR LET?.TXT

 Volume in drive C is HARD DISK C
 Directory of  C:\MEMOS

LET1     TXT      1369 12-23-90   3:54p
LET2     TXT      1404 12-23-90   4:09p
LET3     TXT       509 12-23-90   4:11p
LET4     TXT        62 12-23-90   4:18p
        4 File(s)     104448 bytes free

C:\MEMOS>DEL LET?.TXT
```

The second command line is not retyped. You use the DOS editing keys to modify the original command. Press F1 to display the **D**, and then type **EL** and press F3. The rest of the command line is then displayed.

Using the Root Name of a File in a DIR Command

One anomaly of the DIR command compared with the DEL or COPY commands occurs when you use the root name of a file. If you type **DIR LET1**, the command acts on a different set of files than the **DEL LET1** command. The DIR command will list all files with the root name LET1, those with an extension and those without. The DEL command will delete only those files without extensions. If you add the period that acts as the separator between the root and extension of the file name, the anomaly disappears and the commands behave in a similar fashion.

Caution:
DIR and DEL may act on different groups of files when you use the root name in those commands.

Using DIR Across Subdirectories

Cue:
The DIR command can be used across subdirectories with ease.

The DIR command can be used across subdirectories with ease. Consider a hard disk (drive C) with a subdirectory for a word processor (called WP) that has several subdirectories, two of which are called MEMOS and LETTERS. If you are in the MEMOS subdirectory and want to see the contents of the LETTERS subdirectory, you can type any of the following commands:

```
DIR C:\WP\LETTERS\*.*

DIR \WP\LETTERS

DIR ..\LETTERS
```

Because LETTERS is a subdirectory, DOS understands that when you ask for a directory of LETTERS in the second and third commands, you mean its contents and not just the listing that it is a directory. When you issue a DIR command with a path specifier but no file specifier, DOS knows that you want to see all files in the specified directory. Because DOS works this way, you can omit the *.* portion of the command. The .. selection in the final version refers DOS to the parent directory and down the tree structure to the LETTERS subdirectory.

DOS is not as versatile as it might be because it doesn't include a file finding facility. This facility would search each of the directories in turn to locate a specified file. However, with some care, the DIR command can be used efficiently, reducing the need for other programs to enhance the command. If you have DOS V4, you can use the DOS Shell to arrange lists of files in order to make finding files easier. (See Chapter 16 for more information about the DOS Shell.)

Accessing the DOS Shell User Interface with DOSSHELL

Cue:
The DOS Shell allows you to select items from a list of options rather than having to learn each of the individual commands.

With V4, DOS includes an optional graphical user interface. This interface allows you to select items from a list of options rather than having to learn each of the individual commands. The interface is advantageous to new users because they are able to perform basic DOS functions without extensive training.

This user interface is actually a program known as the *DOS Shell*. You can run the program from the DOS prompt, or you can let DOS run it automatically at boot time from the AUTOEXEC.BAT file. Chapter 16 describes the various features of the Shell program.

Issuing the DOSSHELL Command

The syntax for starting the DOS Shell program is

*d:path***DOSSHELL**

DOSSHELL is an external command, which actually acts as a batch file. Normally, you would need to include the drive and path name with the command. If DOS V4 has been installed in the standard manner, however, the PATH setting will include the DOS directory where the DOSSHELL batch file is located. Therefore, you won't need to give the drive and path name with the DOSSHELL command.

Cue: *When you invoke the DOSSHELL batch file, it calls the Shell program and passes the various parameters to the program.*

The DOSSHELL batch file is created when DOS is installed. The large number of options used when the Shell program is started are included in the batch file. Chapter 16 explains these options in detail.

Understanding the Operation of DOSSHELL

When you invoke the DOSSHELL batch file, it calls the Shell program and passes the various parameters to the program. These parameters specify the mouse type, the screen colors, and many other options.

Although the Shell program is excellent for handling the basic functions of DOS, not all commands are included in the Shell. In addition, for the commands included, all the options are not necessarily available. The command prompt is still necessary for more advanced use of DOS.

Caution: *Not all commands are included in the Shell and, for the commands included, all the options are not necessarily available.*

Changing the Command Prompt with PROMPT

Unless altered, the command prompt shows the current drive and a greater-than sign. The PROMPT command allows you to change from this display to a more informative and friendly prompt.

The most basic form of the PROMPT command was introduced in Chapter 6. Typing **PROMPT pg** at the command prompt changes that prompt to give you the full path on the disk. This new prompt allows you to see the current directory at all times. If you have a hard disk, this display is the minimum prompt setting recommended. Navigating subdirectories is very difficult if you don't know your current position.

Cue: **PROMPT pg** *is the minimum prompt setting recommended if you have a hard disk.*

The PROMPT command has many more features than this. In its most extensive form, the command requires the ANSI.SYS device driver. That function is dis-

cussed in Chapter 12. However, even without using the ANSI.SYS device driver, many different command prompts can be chosen.

Issuing the PROMPT Command

The syntax for the PROMPT command is

PROMPT *string*

The *string* consists of text and/or a series of character pairs. Each character pair, called a *meta-string*, consists of a dollar sign followed by one of the following characters:

t d p v n g l b q h e _

Cue:
The string in a PROMPT command can contain any text or any number of meta-strings in any order.

The string can contain any text or any number of meta-strings in any order. Table 8.1 lists the different meta-strings.

Table 8.1
Meta-Strings for Use with the PROMPT Command

Meta-string	*Displayed information or result*
$_	Carriage return/line feed (moves the cursor to the beginning of the next line)
$b	Vertical bar character (|)
$d	Date
$e	Esc character
$g	Greater-than sign (>)
$h	Backspace (moves the cursor one space to the left)
$l	Less-than sign (<)
$n	Current drive
$p	Current path
$q	Equal sign (=)
$t	Time
$v	DOS version

Understanding the Operation of PROMPT

The **PROMPT pg** command found in Chapter 6 shows the current path followed by the greater than sign. Consider, for example, the following command:

PROMPT Date: $d Time: t_pg

This command displays the following prompt:

```
Date: Mon 09-18-1989 Time: 2:35:07.23p
C:\WP\MEMOS>
```

Two of the meta-strings require further explanation. The Esc character (**$e**) is used in association with the ANSI.SYS driver (explained in Chapter 12). In the same way that the dollar sign is used to indicate to DOS that the next character is a meta-string, the Esc character is used to signal ANSI.SYS that the next few characters are an ANSI.SYS command.

You can use the Backspace character (**$h**) to remove characters from the prompt. For example, in the preceding PROMPT example, the seconds and hundredths of a second in the displayed prompt are more of a distraction than helpful. You might, therefore, alter the command as follows:

PROMPT Date: $d Time: thhhhhh_pg

The result is the following improved prompt:

```
Date: Mon 09-18-1989 Time: 2:35
C:\WP\MEMOS>
```

If you don't like the current prompt, type **PROMPT** alone at the command prompt, and DOS will reset the prompt to the current drive and a greater than sign (for example, `C>`). Keep in mind that if you are using subdirectories, the best practice is to use **PROMPT pg** as the minimum prompt.

Changing the Character Set with CHCP

Cue: You can use different character sets on your PC once the system is configured appropriately.

Depending on your installation, you may be able to select an alternate character set for display. For example, if you need to type in German, the American character set is not adequate. It does not include the various accents and ligatures used. You can use different character sets on your PC once the system is configured appropriately. These character sets on the PC are known as *code pages*. Chapter 15 explains the CONFIG.SYS settings required for using different code pages.

To use more than one code page on a particular computer requires two steps. First, the code pages need to be initialized during the boot process. Second, each code page is activated as required. Only one code page is used at a time, although a computer can be configured to have more than one available for use.

This chapter introduces the change code page command (CHCP). CHCP allows you to change between installed code pages. Chapter 15 shows you how CHCP fits into a general discussion of international considerations for character representation.

Issuing the CHCP Command

The syntax for the CHCP command is

CHCP

or

CHCP *nnn*

nnn is the reference number for the desired code page.

Understanding the Operation of CHCP

If you type **CHCP** alone at the command prompt, the current code page number is displayed. This version of the command works even if you haven't configured your computer for alternate code pages. With the default North American code page, DOS returns the following message:

```
Active code page: 437
```

Code page 437 is an integral part of the hardware in the PC. To change the active code page from the more common 437 to the multilingual code page 850, type

CHCP 850

Caution: *The code page will be changed only if the proper initialization has been done.*

The code page will be changed only if the proper initialization has been done. If not, DOS displays the error message `NLSFUNC not installed`.

Altering the Look of the Screen with MODE

Cue: *Use the MODE command to change the appearance of the display.*

The external MODE command is used extensively with DOS to customize the standard configuration. This command sets the modes of printers, the code page switching, as well as the display. This chapter introduces the MODE command as it can be used to change the appearance of the display. Chapter 15 covers the MODE command options that deal with code pages, and Chapter 9 describes the options associated with devices.

You can have two different types of displays attached to the same computer: a monochrome display adapter and display, and a color graphics adapter and display. You switch between the displays at will using the MODE command. Being able to switch displays is particularly handy when you use graphics-based programs such as a drawing program, but also with text-based programs such as a word processor or database program. MODE allows you to use the color display to show the graphics and drawing you are working on, and the monochrome display to show the text-based programs.

When you type at the keyboard on a two-display system, you see only the keystrokes on one of the displays. That display is referred to as the *active display*. The keyboard and active display together constitute the console. DOS uses the device name CON for the console.

Issuing the MODE Command

Cue: *Two forms of the MODE command can be used with any version of DOS to change the display characteristics; two additional forms are available with DOS V4.*

Two forms of the MODE command can be used with any version of DOS to change the display characteristics. Two additional forms are available with DOS V4.

The syntax for the simplest form of the MODE command, which can be used with any version of DOS, is

MODE *dt*

dt is the display type and mode. Available options are 40, 80, BW40, BW80, CO40, CO80, or MONO.

The 40 and 80 refer to the number of columns of text to be displayed. BW stands for black and white, CO stands for color, and MONO refers to the monochrome display adapter.

Understanding the Operation of MODE

To set the number of columns displayed on the active display to 40 or 80, you type, respectively, either

MODE 40

or

MODE 80

To select the display attached to the monochrome display adapter as the active display, you type

MODE MONO

The monochrome display always displays 80 columns.

To select the color graphics display as the active display, you use one of the following commands:

MODE CO40

MODE CO80

MODE BW40

MODE BW80

The first two options select the color mode of the color display with 40 or 80 columns, as selected. The last two options select the color display, but use the black and white display modes with 40 or 80 columns, as appropriate.

Using MODE on a Color Graphics Adapter

The second MODE command form, available with all versions of DOS, is for use on a Color Graphics Adapter only. This form does not work on an Enhanced Graphics Adapter (EGA) or a Video Graphics Array (VGA). This MODE command form, originally developed to allow users to configure the PC to work with a television instead of a specially designed computer monitor, moves the horizontal position of the displayed image on-screen. If you find that you can't see the leftmost or rightmost character on the screen, this command will help.

The syntax is

MODE *dt,s,T*

The *dt* parameter is optional but can have any of the forms described for the first MODE command form except for MONO. If omitted, the current display and number of columns are maintained. *s* is either an **R** to move the image to the right or an **L** to move the image to the left. If the display is in 80-column mode, the image is moved two characters right or left. If in 40-column mode, the image is moved one character. The optional *T* parameter, when used, causes a test pattern to be displayed, which you can use to align the display.

If you type **MODE CO80,R,T**, you will see a single line of 80 characters displayed across the screen followed by a prompt asking

```
Do you see the leftmost □? (y/n)
```

This image is moved two positions to the right of its previous position. If you respond no to the prompt, the screen is moved further to the right. Using the **L** option works in the same way, but moves the image to the left.

Note that if you decide to operate your system with the displayed image in a shifted position, you will have somewhat less memory available for your application. (The MODE command uses less than 1K of memory.) To display the image in a nonstandard position, DOS puts a small program into memory. That program intercepts everything to be displayed, adjusts it, and then sends it to the display. The program left in memory that changes the system in some way is referred to as the *memory-resident portion* of the MODE command. Some other versions of the MODE command also leave a portion resident.

Caution:
If you decide to operate your system with the displayed image in a shifted position, you will have less memory available for your application.

Using MODE To Adjust the Number of Columns or Lines On-Screen

DOS V4 introduced two additional MODE command options, which are used to alter the display modes. You can adjust the number of columns or the number of lines displayed on-screen. The ANSI.SYS device driver must be installed for both these forms in order to adjust the display.

Cue:
Use the MODE command in DOS V4 to adjust the number of columns or the number of lines displayed on-screen.

Adjusting the Number of Columns

The syntax for the first form of the MODE command that can be used only with DOS V4 is

MODE CON COLS=*a* **LINES=***b*

COLS= sets the number of columns displayed on the console to *a*, and **LINES=** sets the number of lines displayed on the console to *b*. If you omit a setting for the number of columns or the number of lines, the current setting is preserved.

Valid numbers for *a* are 40 or 80. Valid numbers for *b* on a VGA are 25, 43, or 50; and for b on an EGA, 25 or 43.

Provided that ANSI.SYS is loaded through the CONFIG.SYS file, typing the following command results in the display mode being 80 columns wide and 43 lines high on a computer with an EGA or VGA adapter and monitor:

MODE CON COLS=80 LINES=43

After you have set the display mode, you then can start your application program. The display mode, unless reset by the application, will remain. For example, using the preceding MODE example on an EGA-based computer would allow you to use WordPerfect in 43-line mode without adjusting the settings in WordPerfect.

Not all applications will sense the extra lines available. You will have to try some **MODE CON** commands to see whether you can use the extra lines. Using **MODE CON** to set 43 or 50 lines makes viewing long DIR listings more convenient than using the standard 25 lines.

Adjusting the Number of Lines

The final MODE command option added in DOS V4 to adjust the display mode allows you to alter the number of lines displayed on-screen without specifying that the screen is the console. This form, which is really only a variation on the **MODE CON** theme, is handy when you are using an auxiliary console instead of CON. Chapter 9 introduces the CTTY command used to establish another device as the standard input and output device.

The syntax for this form of the MODE command is

*d:path***MODE** *dt,b*

dt is the display type, and *b* is the number of lines to be displayed. The acceptable values for *dt* and *b* are as previously described. Note that not all combinations of values for *dt* and *b* are possible. For example, you can't adjust the number of lines on a monochrome monitor or the CGA.

On a VGA system, you alter the display type to color with 40 columns and 50 lines by typing

MODE CO40,50

All the forms of the MODE command that adjust the display are similar in syntax and purpose. Issuing an incorrect command will not damage anything. DOS provides reasonably clear error messages. For example, if the ANSI.SYS driver is required and not installed, the following error message is displayed:

```
ANSI.SYS must be installed to perform requested function
```

Starting the Command Processor with COMMAND

Cue:
The command processor, COMMAND.COM, is the portion of DOS that actually interprets what you type at the keyboard and executes the various commands.

As discussed in Chapter 3, the command processor, COMMAND.COM, is the portion of DOS that actually interprets what you type at the DOS prompt and executes the various commands. When you start the computer, a copy of the command processor is automatically loaded into memory and is ready for your use.

A command processor has associated with it various settings that are stored in a reserved area of memory referred to as the *environment*. These settings affect the system. For example, any PROMPT setting is stored in the environment. When you issue a command, the command interpreter executes the command; when the command is completed, the command interpreter looks in the environment for the prompt setting and then redisplays the appropriate command prompt.

Other settings stored in the environment include the location of the command interpreter itself on a disk, any path setting, and the prompt setting. If you have enabled **APPEND /E**, the APPEND path is stored in the environment as well. These settings are established in the AUTOEXEC.BAT and CONFIG.SYS files discussed in Chapters 13 and 14, respectively.

You can start a second command processor by using the COMMAND command. This second command processor loads a new copy of the command interpreter into memory. Commands you type at the DOS command prompt are then processed by the second command processor.

Cue:
Use COMMAND to start a second command processor.

The programming techniques involved in creating your own command processor are beyond the scope of this book. However, there is a possibility that you may obtain one with another program. If you do, you invoke it with the COMMAND command.

Issuing the COMMAND Command

The syntax for COMMAND is

COMMAND *d:path*\ */P /MSG /E:aaaaa /C string*

d:path\ is the drive and path position of the COMMAND.COM file to be loaded.

Understanding the Operation of COMMAND

If you omit the *d:path*\ parameter, the second command processor inherits the contents of the first command processor's environment. If you include that parameter, the second command processor does not inherit the old environment and keeps only the COMSPEC path specified by *d:path*\.

Cue:
If you omit the d:path\ parameter, the second command processor inherits the contents of the first command processor's environment.

When you leave the second command processor (via the EXIT command, described later in this chapter), the first command processor's environment is unchanged even if the second command processor's environment was changed. Any environment variable assignments made during the use of the secondary command processor are lost when you exit.

The optional */P* switch makes the second command processor permanent. The first command processor is no longer available, and DOS runs the AUTOEXEC.BAT file, if present. The EXIT command cannot be used to exit this second command processor and return to the first. The computer must be rebooted to return to the first command processor.

Caution:
The computer must be rebooted to return to the first command processor if the secondary command processor was started with /P.

When used, the optional */MSG* switch causes DOS to load DOS messages into memory instead of reading them from disk each time they are needed. In a floppy disk system, using this switch will improve the performance of DOS, but at the cost of losing some memory.

Caution: *Because each character stored in the environment occupies space, you may find that you run out of room in the environment.*

The optional */E:aaaaa* switch allows you to adjust the number of bytes of memory set aside for the environment. The minimum value for *aaaaa* is 160, and the maximum is 32768. Each character stored in the environment occupies space, and you may find that you run out of room in the environment. For example, the environment needs to store long settings such as the long prompt setting shown in the last example about the PROMPT command. Adjusting the environment size may be necessary in some situations.

The */C string* option allows you to pass a string of characters to the command processor you are starting. This option is generally used in association with batch files (see Chapter 13).

Using EXIT To Leave the Current Copy of the Command Processor

Cue: *Use the EXIT command to leave the current copy of the command processor and return to the previously loaded copy.*

Use the EXIT command to leave the current copy of the command processor and return to the previously loaded copy.

Issuing the EXIT Command

The syntax for the EXIT command is

EXIT

Caution: *You can't use the EXIT command if the second command processor was started using the /P switch.*

No options exist for this command. The command cannot be used if the second command processor was started using the */P* switch.

Starting and Exiting Secondary Command Processors

The COMMAND and EXIT command pair has two interesting uses. When you have specified an alternate location with the file COMMAND.COM, you use the COMMAND command to load the second processor. For example, if the DOS command processor is in the root directory and another command processor is located in the OTHER subdirectory, you can type

COMMAND C:\OTHER /E:320

This command loads the different COMMAND.COM and assigns it an environment size of 320 bytes. You can use the second command processor to execute commands using the 320-byte environment. When you are finished using the secondary command processor, type **EXIT**. This command takes you back to the primary command processor again.

The other use for this command pair is when you have set up a complicated environment, and you want to execute a command with a basic environment without changing the existing environment. In this case, you can start the second command processor with the *d:path*\ option. If necessary, change the environment by using the SET command, execute the desired commands in the altered environment, and then exit the second command processor.

Cue: *When you have a complicated environment set, use EXIT to execute a command with a basic environment without changing the existing environment.*

If the command processor is in the root directory of drive C, type

COMMAND C:

This command loads a second copy of the command processor. The new environment will include only a setting for COMSPEC showing that the command processor is located in C:\. The prompt will not have a setting and will have the standard form of `C>`.

You then execute any desired commands in the new environment. When the commands are completed, type

EXIT

This command exits the second command processor and returns you to the first command processor. All the environment settings for the first command processor are preserved.

The principles used by DOS in executing this command are also used by applications programs that allow you to suspend your program temporarily and go to DOS. When you select the DOSSHELL command within your word processor, for example, the program starts a second command processor. All the existing information is kept in memory with the first command processor. You execute DOS commands in a second command processor and type **EXIT** when the commands are complete. The first command processor is then active, and you are returned to your application program at the same point as before.

Changing the Environment with SET

Regardless of whether you have a second copy of the command processor loaded or not, you can adjust the environment using a variety of methods. The PATH command, discussed in Chapter 6, allows you to change one of the variables stored in the environment. The prompt setting is also an environment variable that is often set with the PROMPT command.

Cue: *Use the SET command to show the current settings in the environment, or to add or change environment variables.*

DOS includes a more universal command, SET, for altering environment variables. This command can show the current settings in the environment, or the command can add or change environment variables.

Issuing the SET Command

The syntax for the SET command is

SET *name=string*

SET can be typed at the command prompt without any variables. The command then lists all the current settings for environment variables.

name= is the name of the environment variables. Familiar names are COMSPEC, PROMPT, and PATH. However, you can choose your own variable names in addition to these three.

string is the value you want to set the variable to. In the case of PATH, the *string* would be the list of directories you want DOS to search through to find program files.

If you use the *name=* parameter without a value for *string*, the variable specified is a null value (contains nothing).

Changing the Environment Variables with SET

Suppose that you want to use the SET command to change the environment variables. The following three commands change the COMSPEC, PROMPT, and PATH:

```
SET COMSPEC=C:\SYS
SET PROMPT=$p$g
SET PATH=C:\;C:\DOS;C:\SYS;
```

The first command tells DOS that the command processor, COMMAND.COM, is located in the SYS subdirectory of drive C. The second command sets the prompt to include the current path and the greater than sign. The third command puts the root directory, the DOS directory, and the SYS directory of drive C onto the path.

Warning:
The system will fail if your command processor is not located in the root directory of the boot drive.

The prompt and path can be set by using the specific commands. If your command processor is not located in the root directory of the boot drive, you must include a **SET COMSPEC=** command to the AUTOEXEC.BAT file that points to the command processor. If you don't, your system will fail when you leave an applications program that needs to reload the command processor and the system can't find it.

Setting Your Own Environment Variables with SET

The SET command also can be used to set your own variables in the environment. These variables usually are the names of directories or switches used by programs. The program knows to look for a particular variable in the environment and will take the value and use it in the program.

For example, a word processing program may look for a dictionary file called DICT in the current directory. However, if you use the SET command, a different directory can contain the dictionary file. During its installation, the program probably will insert a SET command into the batch file used to invoke the program. This command would have the following form:

```
SET DICT=C:\WP\DICT
```

This command allows the program to look in the WP directory for the dictionary file, instead of looking in the current directory. Keep in mind that setting a variable in the environment is useful only to programs that know to look for that variable.

Each variable stored in the environment occupies space. Programs that need many variables set may require that you increase the area of memory set aside for the environment when DOS is booted. You do this by including the **SHELL=** command in CONFIG.SYS (see Chapter 14).

Cue: Programs that need many variables set may require that you increase the area of memory set aside for the environment when DOS is booted.

Summary

This chapter introduced the commands used to view and alter the DOS system information. Some of the commands, such as VER, are used infrequently. Others, such as DIR, are used in daily operation. The true power and versatility of DOS can be exploited by an extensive knowledge of these commands. The following key points were covered in this chapter:

- The DATE and TIME commands are used to alter the date and time used by DOS when saving files to disk.
- The VER command displays the current booted version of DOS.
- The MEM command provides extensive information about the memory, programs, and devices currently running on your computer.
- The DIR command is one of the most commonly used DOS commands. It provides a listing of the files in a directory or on a disk. The available options allow the listings to be presented in different forms.

- The DOSSHELL command, available with DOS V4, invokes the batch file that is used to load the DOS Shell. This program has a graphical user interface that allows you to select DOS commands and choose programs from a list.
- The PROMPT command alters the command prompt presented for you to enter commands. On systems using hierarchical directories, changing the prompt setting to **PROMPT pg** is advisable. The current path is then always displayed.
- The change code page command, CHCP, allows you to switch between available code pages on your computer. The code pages must be installed before this command will operate.
- The MODE command can be used to alter the appearance of the screen. You can alter the number of lines, number of columns, and current display.
- A second command processor can be invoked with the COMMAND command. Depending on the options selected, the environment variables can either be inherited or reset in the second command processor.
- The SET command allows individual environment variables to be changed. These variables include PROMPT, PATH, and COMSPEC. Other variables can be defined for use by programs.

Chapter 9 covers the commands used when working with devices. You learn how to control input and output devices. That chapter completes the coverage of the DOS core commands.

9

Gaining Control of Devices

Chapter 4 introduces the concept of peripheral devices as they are used with a personal computer. These peripherals enable the user to interface with the computer itself. Devices can supply input to the computer, receive output from the computer, or both accept and receive information. Examples of devices include the hard disk drive, printer, video display, keyboard, and modem.

This chapter covers the commands that enable you to control the devices which are connected to your computer. The commands are used to perform such functions as redirection, selection of alternate keyboards, or altering the names assigned to parts of your hard disk.

The commands covered in this chapter include the following:

- CLS
- GRAPHICS
- PRINT
- ASSIGN
- JOIN
- SUBST
- KEYB
- CTTY
- MODE
- MORE
- FIND
- SORT

Clearing the Screen with CLS

Cue:
Use the CLS command to clear the screen display.

The internal command CLS is used to clear the screen. The command causes all text visible on-screen to be removed. The prompt is then displayed so that you can continue to issue DOS commands.

Issuing the CLS Command

The syntax of CLS is as follows:

CLS

No switches are available with CLS.

Understanding the Operation of CLS

CLS clears only the display on the standard output device. If you have a system with two displays, CLS clears only the active display and not both screens.

Reminder:
Clearing the screen does not change the mode of the display.

Clearing the screen does not, however, change the mode of the display. If, for example, you are operating in 40-column mode (you have issued the command **MODE CO40**), the screen is cleared when you type **CLS**; the prompt is then redisplayed in 40-column mode. Any attributes you may have set are also retained. If you have defined a background color and foreground color using ANSI.SYS escape sequences, these colors are retained.

Using CLS To Clean Up After Batch Files

Reminder:
The CLS command is used most frequently in batch files.

The CLS command is used most frequently in batch files. By inserting a CLS command at the end of your AUTOEXEC.BAT file, for example, you can remove all the superfluous messages that memory-resident programs may display. The command shouldn't miss any error messages, however. Most applications programs give clear error messages and pause so that you can rectify the problem if they fail to load properly. The CLS line in the batch file will not interfere with this process. If, for example, a CHKDSK command included in your batch file finds lost clusters on your disk, CHKDSK pauses and displays the following message:

```
Convert lost chains to files (Y/N)?
```

For more information on the CLS command, see the Command Reference, as well as Chapters 11 and 13, which deal with batch files.

Printing Graphics Screens with GRAPHICS

Cue:
The GRAPHICS command enables you to use the Print Screen key to print graphics screens.

You use the Print Screen (PrtSc on some keyboards) key to print the contents of the display on a printer. This method, however, works only when the screen is in text mode. If you execute the external command GRAPHICS, you can use the Print Screen key to print graphics screens as well. When you execute GRAPHICS, a portion of the program remains memory-resident. Then, when you press Print Screen (or Shift-PrtSc), any necessary conversion is performed before the information is sent to the printer.

Issuing the GRAPHICS Command

The syntax for GRAPHICS with DOS V3.3 and earlier versions is as follows:

GRAPHICS *printer /R /B /LCD /PB:name*

printer is the type of printer to be used.

The */R* switch forces the printout to print black as black and white as white.

The */B* switch prints the background color.

/LCD causes the printbox size to be set for the PC Convertible liquid crystal display size.

/PB:name sets the printbox size to be STD when you specify **/PB:STD**; the switch sets the printbox size to the smaller liquid crystal display size when you specify **/PB:LCD**.

For DOS V4, one additional option is available. The syntax for DOS V4 is as follows:

GRAPHICS *printer d:path\filename /R /B /LCD /PB:name*

d:path\filename is the drive, path, and file name of a printer profile file.

Understanding the Operation of GRAPHICS

The following basic printer types are supported with the *printer* option:

COLOR1
COLOR4
COLOR8
GRAPHICS
THERMAL
GRAPHICSWIDE (DOS V4 only)

Table 9.1 lists the product names that refer to the various printer settings. If your printer is not listed, it may be compatible with one of these products. Refer to the printer instruction manual for details.

Table 9.1
Product Names and Printer Settings

Printer Type	*Model Name*
COLOR1	IBM 5182 Color Printer with black ribbon which prints in gray scales
COLOR4	IBM 5182 Color Printer with RGB (red, green, blue) ribbon which produces four colors
COLOR8	IBM 5182 Color Printer with CMY (cyan, magenta, yellow, and black) ribbon, which produces eight colors
GRAPHICS	IBM 5152 Graphics Printer Model 2 IBM 4201 ProPrinter and ProPrinter II IBM 4202 ProPrinter XL (8 1/2-inch paper) IBM 4207 ProPrinter X24 IBM 4208 ProPrinter XL24 (8 1/2-inch paper) IBM 3812 Pageprinter IBM 5201 Quietwriter II IBM 5202 Quietwriter III
THERMAL	IBM 5140 PC Convertible Printer
GRAPHICSWIDE	IBM 5201 Quietwriter II (13 1/2-inch paper) IBM 4202 ProPrinter XL (13 1/2-inch paper) IBM 4208 ProPrinter XL24 (13 1/2-inch paper)

Cue:
The GRAPHICS printer type is the default with the GRAPHICS command.

If no printer is specified, DOS assumes the GRAPHICS printer type.

When you include the */R* switch, the printed image is *reversed*—black on the screen is printed as black on the paper. On a monochrome monitor, where the text is green in color, DOS treats the green as white.

If you do not use the /R switch, everything on-screen that is white is printed as black, and everything black on-screen (usually the background) is printed as white. The paper color in the printer is assumed to be white.

Use the */B* switch when the COLOR4 or COLOR8 printer type is specified. The switch causes the *background* color to be printed. If you omit the /B switch, the background color is not printed.

The liquid crystal display on the IBM PC Convertible is smaller than that of a full-sized display. With this machine, you need to use one of the following two switches: */LCD* or */PB:name*.

Use */LCD* to get a printed image as on the PC Convertible's liquid crystal display. (Fewer lines are displayed.)

Use */PB:name* also with the PC Convertible. When a full-sized display is attached to the PC Convertible, you use this switch to select the type of printed image. Typing **/PB:STD** causes the printbox size to be that of a full-sized (standard) display. Typing **/PB:LCD** has the same effect as **/LCD** and prints a screen image with fewer lines.

Use the DOS V4 option *d:path\filename* to specify a printer profile file. The optional drive, path, and file name specify a file that contains information about a printer. This file, known as a *profile*, supports the printers of other manufacturers.

Cue:
You can select the name of a printer profile file when using DOS V4 GRAPHICS.

The profile file is an ASCII text file that consists of two sections for each printer in the file. A profile can include information on many printers. One section defines how the printer is controlled, such as how to select each color on the printer, or how to adjust the darkness of the printing. The second section lists the translation from the screen to the printer, such as how large the screen image is to be printed on the printer.

GRAPHICS.PRO is the profile file supplied with DOS. If you are interested in creating a profile file for your printer, make a copy of the supplied GRAPHICS.PRO and try modifying where necessary. If you are interested in the programming aspects of DOS, you can try this exercise. Modifying the GRAPHICS.PRO file is not necessary for most printers.

As with previous versions of DOS, if no printer is specified on the GRAPHICS command line, DOS assumes the GRAPHICS printer type. With DOS V4, if you are using other parameters, but are not specifying a printer type, use a comma to separate the printer definition from the following profile.

Understanding the General Rules of GRAPHICS

- Once GRAPHICS is loaded, pressing the Print Screen (Shift-PrtSc) key allows graphics screens to be printed.
- GRAPHICS allows graphics screens to be printed on supported graphics printers only.
- If the printer cannot print graphics, GRAPHICS will not affect the output seen on the printer.
- You must run GRAPHICS before you select the screen to be printed.

- The name given in the GRAPHICS command must match the first parameter of the printbox statement found in the designated printer profile.
- If you omit /PB and /LCD switches, GRAPHICS uses the previous printbox setting.
- You can print up to 8 colors on a color printer.
- You can print up to 19 shades of gray on a black and white printer.

Cue:
You can use the GRAPHICS command to print up to 8 colors on a color printer, and up to 19 shades of gray on a black and white printer.

Using GRAPHICS To Print a Screen Image

Suppose that you have an IBM ProPrinter printer. If you have a monochrome system and want to print a screen image with the background as black and the text as white, type the following command:

GRAPHICS /R

Once GRAPHICS is loaded into memory, you can create the screen of interest and press Print Screen to print to the printer.

To print eight-color images, including the background color on an IBM 5182 Color Printer with a CMY ribbon installed, use the following command:

GRAPHICS COLOR8 /B

On the PC Convertible with a full-sized monitor attached, you can output the screen image to the IBM 5140 PC Convertible Printer by installing GRAPHICS with the following options:

GRAPHICS THERMAL /PB:STD

Changing the command to either **GRAPHICS THERMAL /PB:LCD** or **GRAPHICS THERMAL /LCD** will print the image as you would see it on the liquid crystal display.

Cue:
Include the GRAPHICS command in your AUTOEXEC.BAT file to make sure that the command is loaded upon booting.

Some early programs may not print properly on your PC. Try issuing a suitable GRAPHICS command before running the program to see whether this fixes the problem. You can also include the GRAPHICS command in your AUTOEXEC.BAT file to ensure that the command is loaded upon booting.

Printing in the Background with PRINT

In Chapter 4, you learn that the COPY command can be used to transfer information from one device to another. For example, the command

COPY LETTER.TXT PRN copies the file named LETTER.TXT to the device PRN, the printer. While this copying process occurs, the computer is not available for other use.

The external command PRINT enables you to print "in the background" so that you can continue to do other work while the printing is being performed. The printing occurs during the idle times, when the computer is waiting for you to type at the keyboard, for example. More than one file can be queued for printing at a time.

Cue:
Use the PRINT command to print in the background.

Issuing the PRINT Command

The syntax for the external PRINT command is as follows:

PRINT */D:device /B:bufsiz /M:maxtick /Q:maxfiles*
/S:timeslice /U:busytick d1:path1\filename1 /P /T /C
d2:path2\filename2 /P /C /T ...

d1:path1\filename1 is the first drive, path, and file name of a file to be printed. *d2:path2\filename2* is the second drive, path, and file name of the file to be printed. The ellipsis (...) signify that more files can be specified.

Wild cards are permitted in the file names.

The optional switches are listed in table 9.2.

Table 9.2
PRINT Switches

Switch	*Action*
/D:device	Specifies the *device* name for the output device.
/B:bufsiz	Specifies the size of *buffer* used in the printing process.
/M:maxtick	Specifies the *maximum* number of system clock *ticks* that the PRINT command can use at a time.
/Q:maxfiles	Specifies the *maximum* number of files that can be *queued* at a time.
/S:timeslice	Specifies the proportion of *time* allocated to the background printing.
/U:busytick	Determines how many ticks the PRINT command waits for the printer to not be *busy*.

Table 9.2—*continued*

Switch	*Action*
/P	Places the file in the line for printing; places the command in *print* mode.
/C	*Cancels* the background printing of the file specified.
/T	*Terminates* all printing and clears the print queue.

Understanding the Operation of PRINT

Reminder:
Both GRAPHICS and PRINT remain in memory once activated.

Like GRAPHICS, PRINT leaves a portion of itself in memory once it is activated. Once installed, the method of operation cannot be changed. As a result, the /D:device, /Q:maxfiles, /B:bufsiz, /S:timeslice, /U:busytick, and /M:maxtick optional switches can only be specified the first time the PRINT command is issued.

The optional switches available with PRINT can change the command's operation.

/D:device specifies the device name for the output device. Acceptable values include any DOS output device, such as LPT1, LPT2, LPT3, PRN, COM1, COM2, COM3, COM4, or AUX.

/Q:maxfiles specifies the maximum number of files—from 4 to 32—that can be queued at a time. If the switch is omitted, 10 files is the default queue size.

/B:bufsiz specifies the size of buffer used in the printing process. The data for printing is taken from the disk in chunks which are the size of this buffer. Increasing the size causes more data to be taken from the disk at a time. The minimum buffer size is 512 bytes—the maximum is 16K. If the switch is omitted, 512 bytes are set aside for the buffer. Remember that the larger the buffer, the less RAM there is available for applications programs.

/S:timeslice determines the proportion of time allocated to the background printing. Too high a value for this switch will cause the computer to give you a sluggish response to other commands executed while printing is occurring in the background. A low value will slow down the printing process. The range of values is 1 to 255; the default value is 8.

/M:maxtick specifies the number of system clock ticks that the PRINT command can use at a time. The PRINT command will print for a certain number of clock ticks before passing control back to DOS for continuing with the foreground task. (The system clock is discussed with the DATE and TIME com-

mands in Chapter 8.) /M:maxtick can be set to any value between 1 and 255; the default value is 2.

/U:busytick is used to determine how long the PRINT command waits for the printer to not be busy. In most cases, the PRINT command sends data to the printer faster than the printer can actually print it. The printer can store some of this information in its own memory. When this memory is full, it signals to the computer that it is "busy." The command is actually transferring the data in its memory to the print head and printing on the page. When some room becomes available in the printer's memory, it stops sending a busy signal, and PRINT can send more data.

The /U:busytick switch can change the number of clock ticks that PRINT will wait to send data. The default setting is 1, but the setting can be altered up to as high as 255. If PRINT tries to send data to the printer and finds it busy, it waits the number of clock ticks set by this switch. If the printer is still busy at the end of this time, PRINT immediately transfers control back to DOS for other tasks without claiming its full time allocation set by the /M:maxtick setting.

Typing just **PRINT** at the prompt will give you a list of files in the queue. This list includes the name of the current file being printed and the order of files waiting to be printed. Error messages are also displayed. If, for example, you forget to turn on the printer, the following error message appears:

```
Errors on list device indicate that it may be
off-line. Please check it.
```

The */P* switch *places* a file in PRINT's queue. The preceding file and all subsequent files on the command line are printed.

The */C* changes the mode to *cancel*. The file name issued before the /C, and all files after the /C on the same command line, are removed from the print queue. You must issue the /P switch to start adding files to the queue again.

If file names are entered without a /P, /C, or /T switch, the /P switch is assumed to have been entered. The files are all placed in the queue for printing.

The */T* switch *terminates* printing. All files are removed from the queue, including the one currently being printed. The printer alarm sounds, a file cancellation message is printed, and the paper is advanced to the next page. This alarm action also occurs when the current file being printed is canceled with the /C switch.

Understanding the General Rules of PRINT

- The /D:device, /Q:maxfiles, /B:bufsiz, /S:timeslice, /U:busytick, and /M:maxtick optional switches can only be specified the first time the PRINT command is issued.

- Once PRINT is installed, /D:device, /Q:maxfiles, /B:bufsiz, /S:timeslice, /U:busytick, and /M:maxtick cannot be changed by issuing the command again.
- If /D:device is specified, it must be the first switch.
- If /T is issued, all files including the one currently being printed are cleared from the print queue.
- If /P is issued, the preceding file and all subsequent files entered on the command line by the PRINT command are printed until a /T or /C is issued.
- If /C is issued, the preceding file and all subsequent files are canceled from the queue of files.

Reminder:
Files are printed in the order in which they are entered at the command line.

- The files are printed in the order entered at the command line.
- A page-eject sequence is sent to the printer at the end of a file.
- PRINT cannot be used on a network server.

Reminder:
You cannot use the printer for other purposes while PRINT is in operation.

- You cannot use the printer for other purposes while PRINT is in operation. You cannot, for example, use Print Screen when PRINT is in effect.
- The files to be printed must remain in the same disk drive and unaltered while they are waiting to be printed and while they are being printed.
- Specifying a nonexistent device causes unpredictable behavior by the computer.
- Tab characters in the printed file are converted to blanks up to the next 8-column boundary.

Using PRINT To Print Several Files

Reminder:
You can specify parameters with PRINT only when the command is first issued.

You do not have to enter all at once the names of the files to be printed. You can issue the PRINT command several times to add or remove files from the print queue. The first time you issue the command, however, is the only time you can specify the various parameters that affect the way PRINT operates. Once resident, the PRINT command is used to enter file names for printing.

Entering the PRINT command for the first time without defining a device causes PRINT to prompt you for a device name. PRN is the default offered. Pressing Enter at the prompt accepts PRN, which is usually LPT1.

Suppose that you have five files for printing called LETTER1.TXT, MEMO1.TXT, REPORT1.TXT, REPORT2.TXT, and REPORT10.TXT. In a standard configuration, accepting PRINT's default setting, you would type the following:

PRINT /D:PRN LETTER1.TXT /P MEMO1.TXT REPORT1.TXT REPORT2.TXT REPORT10.TXT

If you then decide that you want to print REPORT10.TXT before REPORT2.TXT, you type the following:

PRINT REPORT2.TXT /C REPORT2.TXT /P

This command removes REPORT2.TXT from the print queue and adds it to the end of the queue.

You can cancel all files to be printed by typing the following command:

PRINT /T

By using a combination of the /P, /T, and /C switches, you can adjust the order of printing the files.

Cue: *Adjust the default installation settings for PRINT to achieve better response time on your computer.*

If you are not in a hurry to collect the printed output of files, and you are going to use the computer while the printer is printing, you may want to adjust the default installation settings for PRINT. Adjusting the settings will give your computer better response time when you are working on something else.

You can also try adjusting the /M:maxtick or /S:timeslice settings. /M:maxtick alters the length of time that PRINT prints once it gets control, and /S:timeslice determines the number of timeslices that occur between the times PRINT gets control from DOS. Try the following when you invoke the PRINT command for the first time:

PRINT /D:PRN /M:1 /S:100

If you are not worried about the sluggishness of the keyboard, you can improve the speed of the background printing by altering the buffer size as well as adjusting /M:maxtick and /S:timeslice. Consider the following setting:

PRINT /D:PRN /B:16384 /M:90 /S:4

Remember that a system clock tick occurs once every 1/18.2 seconds. This command line gives PRINT control every four clock ticks for a period of approximately five seconds. The buffer size is also increased to its maximum.

Experiment with these variables until you find one that is acceptable to you. A sluggish keyboard is not always tolerable. If the response time is too slow, you can make errors. You can, for example, assume that a program did not accept your keystroke and try to retype the command. The program, for its part, was just waiting to regain control.

Redirecting File I/O with ASSIGN

Cue: *Use the ASSIGN command to redirect I/O requests from one drive to another.*

The external command ASSIGN is used to redirect all DOS input and output (I/O) requests from one drive to another. Each drive is a DOS device; by using the ASSIGN command, DOS can interrogate a different disk than the one actually specified on a command line.

Issuing the ASSIGN Command

The syntax for the ASSIGN command is as follows:

ASSIGN d1=d2 ...

d1 is the drive letter for the original disk drive; **d2** is the drive letter for the reassignment.

The ellipsis (...) indicates that more than one drive can be reassigned on a single command line.

Understanding the Operation of ASSIGN

When you issue the ASSIGN command, DOS redirects all I/O requests for **d1** to **d2**. If you assign drive A: to drive C:, all commands referring to drive A: will actually be sent to drive C:.

The ASSIGN command should be used sparingly. If it is not required for a particular purpose, don't use it. Certain programs may require disk information that is not available from the reassigned drive.

Understanding the General Rules of ASSIGN

- You can use an optional colon after the drive letter with DOS V4 only.
- Remove any ASSIGN settings before running FDISK, PRINT, FORMAT, BACKUP, or RESTORE.
- Do not use ASSIGN if either JOIN or SUBST are being used.
- Do not use ASSIGN prior to using the APPEND command.
- Beware of using CHDIR, MKDIR, RMDIR, LABEL, APPEND, and PATH with any drives reassigned.

- DISKCOPY and DISKCOMP do not recognize any drive reassignments.
- **d1** and **d2** must both physically exist in the computer.

Using ASSIGN To Reassign Drives

Some early versions of programs may not be able to use drives other than drive A: or B:. In many cases, if you reassign drives A: and B: to drive C:, you can run an otherwise uncooperative program on a hard disk.

In some cases, by reassigning a drive, you can avoid having to use a floppy disk to run a program. Certain programs assume that their program floppy disk is located in drive A:. Reassignment of drive A: to the hard disk results in the program being directed by DOS to the hard disk instead of to the floppy drive.

To use the ASSIGN command to assign drive A: to drive C:, you type the following:

ASSIGN A=C

If you now issue a directory command to drive A: or drive C:, you will get a directory listing for drive C:.

Typing just **ASSIGN** at the prompt removes all the reassignments previously made.

Joining Two Disk Drives with JOIN

The ASSIGN command makes an entire disk drive appear to have a different drive letter than its real value. The JOIN command is used to add a disk drive to the directory structure of another disk.

Cue:
Use the JOIN command to add a disk drive to the directory structure of another disk.

The external command JOIN enables you to have a floppy disk that appears to be part of a hard disk. The directory structure on the floppy disk is added to the directory structure of the hard disk. You can also use JOIN if you have two hard disks—drive C: and drive D:. You use the command to attach drive D: to a sub-directory on drive C:.

Issuing the JOIN Command

To connect disk drives, use the following form:

JOIN d1: *d2:***dirname**

To disconnect disk drives, use this form of the command:

JOIN d1: /D

To show currently connected drives, use the following form:

JOIN

d1: is the disk drive to be connected. DOS calls this drive the *guest disk drive*.

d2: is the disk drive to which **d1:** is to be connected. DOS calls *d2:* the *host disk drive*.

dirname is a subdirectory in the root directory of *d2:*, the host drive. DOS calls **dirname** the *host subdirectory*. **dirname** holds the connection to **d1:**, the guest drive.

/D Disconnects the specified guest disk drive from its host.

Understanding the Operation of JOIN

Hierarchical directories and associated commands are discussed in Chapters 2 and 6. The JOIN command reassigns a drive to a subdirectory position on a different disk. Any directory hierarchy on the reassigned disk becomes a part of the other disk's hierarchical structure.

Using the terminology of a directory tree, the JOIN command can be thought of as grafting a second tree onto the first. The second tree is positioned at least one level down the structure. The grafting point must be assigned a name so that DOS can refer to it. In this way, the root directory of the reassigned disk is given a new name. All directories below the root directory on this reassigned drive have this new name as part of their path.

DOS internally converts all I/O requests to this new subdirectory—and all layers below the subdirectory—into a drive assignment with the appropriate path.

Understanding the General Rules of JOIN

- **dirname** must either be empty or not exist.
- You cannot join a directory to the current directory.
- You cannot join to the root directory of a drive.
- When a drive is joined to another, you will not be able to access **d1:**.
- The entire drive is joined with the JOIN command.
- You cannot specify a networked drive either as **d1:** or *d2:*.
- Do not use the JOIN command with ASSIGN or SUBST.

- Remove any JOIN settings before running DISKCOPY, DISKCOMP, FDISK, PRINT, FORMAT, BACKUP, or RESTORE.
- Beware of using CHDIR, MKDIR, RMDIR, LABEL, APPEND, and PATH with any drives reassigned.

Using JOIN To Connect Drives

To JOIN drive B: to drive C: as a subdirectory called DRIVEB, type the following:

JOIN B: C:\DRIVEB

Typing just **JOIN** at the prompt displays any drive reassignments. In this example, the following message appears:

```
B: => C:\DRIVEB
```

To remove a joined drive, type the following:

JOIN B: /D

The JOIN command can be used to overcome the limitations of DOS. The command is not, however, an ideal solution in all cases. Certain commands must not be used when JOIN is in effect. Joining two logical partitions on a single drive overcomes the 32M partition barrier. Choosing to use the JOIN command is a compromise between full DOS functionality and the benefit of a larger partition.

Substituting a Drive Name for a Path with SUBST

The external command SUBST is an inverse of the JOIN command. Instead of grafting a second disk onto the tree structure of another disk, the SUBST command splits a disk's directory structure into two. In effect, the SUBST command creates an alias disk drive name for a subdirectory.

Cue: *The SUBST command splits a disk's directory structure into two.*

Issuing the SUBST Command

The syntax for the SUBST command is as follows:

To establish an alias, use the following form:

SUBST d1: *d2:***pathname**

To delete an alias, use this form:

SUBST d1: /D

To see the current aliases, use this form:

SUBST

d1: is a valid disk drive name that becomes the alias or nickname. **d1:** may be a nonexistent disk drive.

*d2:***pathname** is the valid disk drive name and directory path that will be nicknamed **d1:**.

Understanding the Operation of SUBST

The SUBST command replaces a path name for a subdirectory with a drive letter. Once a SUBST command is in effect, DOS translates all I/O requests to a particular drive letter back to the correct path name.

As the default, DOS assigns the `LASTDRIVE=` parameter a value of E. Higher drive designators, however, can be made by inserting a `LASTDRIVE=` parameter into the CONFIG.SYS file. DOS then establishes each of the drive letters up to and including the specified LASTDRIVE as DOS devices. When you use the SUBST command, DOS understands that you are referring to a device.

The alias drive "created" by the SUBST command inherits the directory tree structure of the subdirectory reassigned to a drive letter.

Understanding the General Rules of SUBST

- **d1:** and *d2:* must be different.
- You cannot specify a networked drive either as **d1:** or *d2:*.
- **d1:** cannot be the current drive.
- **d1:** must have a designator less than the value in the LASTDRIVE statement of CONFIG.SYS. (See Chapter 14 for more information on CONFIG.SYS.)
- Do not use SUBST with ASSIGN or JOIN.
- Remove any SUBST settings before running DISKCOPY, DISKCOMP, FDISK, PRINT, FORMAT, LABEL, BACKUP, or RESTORE.
- Beware of using CHDIR, MKDIR, RMDIR, APPEND, and PATH with any drives reassigned.

Using SUBST To Reference a Path with a Drive Letter

SUBST is commonly used in two different situations. If a program that does not support path names is being used, the SUBST command is used to assign a drive letter to a directory. The program then refers to the drive letter, and DOS translates the request into a path. If, for example, the data for a program is stored in C:\WORDPROC, you can type the following:

SUBST E: C:\WORDPROC

You tell the program that the data is stored in drive E:.

When the substitution has been made, you can issue the following command:

SUBST

The following message will appear:

```
E: => C:\WORDPROC
```

To disconnect the substitution of drive E: for the C:\WORDPROC directory, you type the following:

SUBST E: /D

The other use for SUBST is to reduce typing long path names. Typing long path names can be a tedious process when a PC is used by more than one person. Each user may have a separate section of the hard disk for storing data files, but common areas of the disk are used to store the programs. If the paths \USER1\WORDDATA and \USER1\SPREDATA exist on drive C:, the typing needed to reach files in the directories can be reduced by entering the following command:

SUBST E: C:\USER1

Issuing a directory command on drive E: produces the following listing:

```
Volume in drive E is HARD DISK C
Directory of  E:\

.            <DIR>      06-02-88   2:07p
..           <DIR>      06-02-88   2:07p
WORDDATA     <DIR>      05-01-89  11:59a
SPREDATA     <DIR>      06-02-88   2:08p
       4 File(s)      77824 bytes free
```

The volume label given is the label from drive C:, but the directory itself is the contents of C:\USER1.

As with JOIN and ASSIGN, the choice to use this command is a compromise. In some situations it is beneficial, but you must be aware of the commands not to be used when SUBST is in effect.

Extending a Keyboard's Code Capability with KEYB

Cue:
Use the KEYB command to switch between keyboards and associated code pages.

You use the external command KEYB to switch between keyboards and their associated code pages. Different languages need different character sets; the alternate sets supplied with the PC enable a variety of languages to be used.

The syntax forms of KEYB varies dramatically depending on the version of DOS used. Early versions of DOS can issue the command with no preparation. DOS V3.3 and V4 require that code pages be prepared. Chapter 15 explains these installation procedures.

Issuing the KEYB Command

The syntax for the KEYB command varies with the version of DOS.

For DOS V3.2, you can use the following syntax:

KEYB*xx*

xx refers to UK, GR, FR, IT, or SP.

For DOS V3.3, you can use the following syntax:

KEYB *keycode,codepage,d:path\filename.ext*

keycode is the two-letter key code; *codepage* is the code page; *d:path\filename.ext* is the keyboard definition file.

For DOS V4, you can use the following syntax:

KEYB *keycode,codepage,d:path\filename.ext /ID:code*

or

KEYB *code*

keycode is the two-character keyboard layout name. *codepage* is the three-digit code page number. *d:path\filename.ext* is the drive, path, and file name for the keyboard device driver. *code* is the keyboard identification code.

Typing just **KEYB** displays the current keyboard settings for keyboard code, code page, and the console's code page.

Understanding the Operation of KEYB

Once a keyboard driver is loaded, DOS treats it as a device. As you type, DOS converts the key scan codes to the appropriate characters for the current keyboard device. Switching between keyboards causes DOS to use a different "translation table" to convert what is typed at the keyboard into characters on-screen.

For DOS V3.2 and earlier versions, the **KEYB***xx* command loads the appropriate alternate keyboard. No options are available. Only one keyboard definition exists for each country. You can, however, alternate between the US keyboard and the loaded driver. For DOS V3.3 and V4, more variety is possible. Each keyboard has a different keyboard layout, keyboard ID, and supported code pages. The UK keyboard layout, for example, is available with two different keyboards, each with its own ID. Two code pages are supported on these two keyboards. The keyboard drivers used must correspond with their appropriate DOS versions.

Once various keyboards are installed, the KEYB command is used to switch between them.

The available keyboard drivers for various DOS versions are shown in table 9.3.

Table 9.3
Keyboard Drivers

Drivers for DOS V3.2 and earlier versions

Command	*Keyboard*
KEYBFR	France
KEYBGR	Germany
KEYBIT	Italy
KEYBSP	Spain
KEYBUK	United Kingdom

Drivers for DOS V3.3 and V4

Keyboard layout	*Code Pages*	*Keyboard ID*	*Country*
BE	850, 437	120	Belgium
CF	863, 850	058	Canada (French-speaking)
DK	850, 865	159	Denmark
FR	437, 850	120, 189	France
GR	437, 850	129	Germany
IT	437, 850	141, 142	Italy
LA	437, 850	171	Latin America
NL	437, 850	143	Netherlands
NO	850, 865	155	Norway

Table 9.3—*continued*

Keyboard layout	*Code Pages*	*Keyboard ID*	*Country*
PO	850, 860	163	Portugal
SF	850, 437	150	Switzerland (French-speaking)
SG	850, 437	000	Switzerland (German-speaking)
SP	437, 850	172	Spain
SU	850, 437	153	Finland
SV	437, 850	153	Sweden
UK	437, 850	168, 166	United Kingdom
US	437, 850	103	United States

Understanding the General Rules of KEYB

- The keyboard driver must correspond with the correct version of DOS.
- For DOS V3.3 and V4, the code page must be prepared before using the KEYB command.
- Load KEYB before any program that redefines keys.
- The directive `INSTALL=` can be used to load KEYB from the CONFIG.SYS file instead of the command line.
- With DOS V4, if you do not specify a code page, the currently active CON code page is used.

Using KEYB To Enable Foreign Language Keys

With DOS V3.2, typing the command **KEYBGR** will load the German keyboard layout. You can then use the accented characters appropriate to the German language.

With DOS V3.3 or V4, the following command will load the device driver for the German keyboard and set the multilingual code page (850):

KEYB GR, 850

With DOS V4, the following command selects the United Kingdom keyboard, ID number 168, with the code page 437:

KEYB UK, 437, C:\DOS\KEYBOARD.SYS /ID:168

Selecting a Different Console with CTTY

DOS usually uses the keyboard and screen as the standard input and standard output devices. Together, these two elements comprise the console, known to DOS as the CON device. The internal command CTTY enables you to select a different device as the console and restore the keyboard and screen as the console.

Cue: The CTTY command enables you to select a different console.

Issuing the CTTY command

The syntax for the internal command CTTY is as follows:

CTTY device

or

CTTY CON

device is the name of a DOS device.

Understanding the Operation of CTTY

CTTY causes DOS to intercept the I/O request calls that normally come from the keyboard and go to the screen; the command redirects these calls to an alternate device.

Understanding the General Rules of CTTY

- The character-based devices AUX, COM1, COM2, COM3, or COM4 can be used as the alternate console.
- The physical device attached to the relevant AUX, COM1, COM2, COM3, or COM4 must be able to accept input and provide output.
- Programs that do not use DOS function calls will not make use of the alternate console.
- If you load BASIC, which does not use DOS function calls, the standard input and output are restored.

Making the Serial Port the Console with CTTY

Typing the command **CTTY COM1** makes COM1 the device that will send and receive standard input and output. This command is used in association with specialized programs that need input from a different source to the keyboard.

Another use for the command is possible if the computer is attached to an intelligent bar code reader which is collecting information from packages. This reader, in association with a specialized program, may not need to use the display or keyboard.

The typical user would probably not use the CTTY command. Information can be gathered through these devices without altering the standard input and output devices. Certain applications, however, would benefit from this feature.

Typing **CTTY CON** from the auxiliary device restores the console to the keyboard and display.

Controlling Device Operations with MODE

Cue:
Use MODE to adjust the settings for a variety of devices.

MODE is one of the more versatile external commands supplied with DOS. DOS V4 has extended its functionality still further. MODE is used to set the operational modes of serial ports and parallel ports, and to redirect information from a parallel port to a serial port. Other functions include setting display modes (see Chapter 8) and use with code pages (see Chapter 15).

In this chapter, the MODE functions that relate to devices are introduced. The Command Reference section details all of the MODE functions. MODE can be used to set a variety of devices—the serial ports and parallel ports.

Using MODE To Change Parallel Port Settings

As you learned in Chapter 4, a parallel port transmits data by transferring the entire byte at one time. The MODE command adjusts the number of lines per inch, the number of columns per line, and the retry feature.

Issuing MODE To Change Parallel Port Settings

The syntax for the external command MODE command that changes the parallel printer characteristics is as follows:

MODE LPT*x*:*cpl,lpi, P***

LPT*x*: is the parallel port name; *cpl* is the number of characters per line; *lpi* is the number of lines per inch; and *P* specifies continuous retries on time-out errors.

The default value of *cpl* is 80; the default value of *lpi* is 6.

An additional command mode is available with DOS V4. This syntax is as follows:

MODE LPT*x*: *COLS=wid, LINES=lpi, RETRY=action*

wid is the number of columns per line; *lpi* is the number of lines per inch; and *action* is the message you want DOS to return when it finds the printer port busy. Available actions are as follows:

E	Returns an *error* when the printer port is busy
B	Returns a *busy* signal when the printer port is busy
R	Returns a *ready* signal when the printer port is busy

Understanding the General Rules of MODE To Change Parallel Port Settings

- The default values of the port are reset if the printer is reset or initialized.
- A parameter omitted from the command line is not changed.
- Do not use the P option when printers are being shared on IBM PC Network.

Understanding the Operation of MODE with Parallel Ports

The MODE command used to adjust the parallel port alters only two items seen on the printout itself—the characters per line and the lines per inch. In general, printing is performed directly from an applications program, which is able to set many more parameters for a particular printer. The program operates by sending Escape sequences to the printer, which can then adjust the printed output accordingly.

The retry setting is more significant. On versions of DOS prior to V4, the retry option either performs no retry when a time-out error is returned, or performs continuous retries when the P option is included. When data is sent to the printer, the port expects to see signals from the printer indicating that the data has been received. If no signals are received within a particular period of time,

a time-out error is generated. By default, DOS will not reattempt to send information to the printer and will return an error message to the screen.

Including the P option prevents this error message from being displayed as DOS continuously tries to send the data. The Ctrl-Break key combination will stop this retrying process.

Cue:
DOS V4 forms of MODE offer additional retry features.

In DOS V4, more retry options are available. With no retry specified, DOS does not continue to try to send data when a time-out error occurs.

When the retry feature is specified, you can select from three options. The B setting is equivalent to the P setting in previous DOS versions and will return a busy signal to the device driver when the port is busy. The R setting will cause a ready signal to be returned from the port even if the port is really busy. Then, when the printer does become ready, the data is ready to send, and an error message will not appear on-screen. The E option is most commonly used when the printing is being done in the background (by PRINT or a network print queue). The data is not transferred to the printer until the port is not busy.

Using MODE To Print a Large File

Cue:
Use the P option with MODE to print a large file.

In certain cases, the P option is necessary to allow a file to be printed. Consider a large DOS file that you want to copy to the printer. If the file is larger than the storage capacity of the printer, the printer port is going to be busy at some point during the data transfer. If the printer remains busy for too long a period of time, an error message appears on-screen; DOS thinks the printer is defective, and the printing process aborts.

Specifying **P** in DOS V3.3 and earlier versions, or **RETRY=B** in DOS V4, will cause DOS to wait until the printer is ready to receive data. The command for LPT1 in DOS V3.3 is as follows:

```
MODE LPT1:,,P
```

For DOS V4, the command is as follows:

```
MODE LPT1 RETRY=B
```

If the file is large, it may also be convenient to have it printed with more lines per inch (8 instead of 6), so that more lines of text can fit on a page. You can also specify 132 columns per line instead of 80. This is not a problem for printers with wide carriages that accept wide paper. For a printer that accepts 8 1/2-inch paper, you will probably need to set the printer to print in condensed print to fit 132 columns on a single line.

The command to fit as much information on a page as possible, when the printer is attached to LPT2:, is as follows:

```
MODE LPT2:132,8
```

Using MODE To Change the Serial Port

Cue:
Use MODE to alter the functions of the serial ports.

Another option of the MODE command is used to alter the functions of the serial ports. This command works in a similar way to the parallel port adjustments. DOS changes the parameters being sent to and from the device driver.

Issuing MODE To Adjust Serial Ports

The syntax for the MODE command that changes the serial port is as follows:

MODE COM*y*: *baud,parity,databits,stopbits,P*

COM*y*: is the name of the serial port device; *baud* is the baud rate; *parity* is the parity; *databits* is the number of data bits; *stopbits* is the number of stop bits; and *P* specifies continuous retries on time-out errors.

The baud rate for the serial port must be specified.

The default number of data bits is 7. The default number of stop bits is 1 for all baud rates except 110, when 2 stop bits are set as the default.

With DOS V4, the following syntax is possible:

MODE COM*y*: *BAUD=baud PARITY=parity DATA=databits*
STOP=stopbits RETRY=action

action is the message you want DOS to return when it finds the port busy.

Available actions are as follows:

E	Returns an *error* when the printer port is busy
B	Returns a *busy* signal when the printer port is busy
R	Returns a *ready* signal when the printer port is busy

Understanding the General Rules of MODE with Serial Ports

- If a retry option is set, a portion of MODE remains resident unless the **RETRY=none** option of DOS V4 is used.
- The retry option slows the performance of foreground tasks when computers are being shared on IBM PC Network.
- The parity settings in DOS V4.0 can include *Mark* or *Space* parity. All versions of DOS support *none*, *odd*, and *even* parity settings.

Understanding the Operation of MODE with Serial Ports

Cue:
The baud rate refers to the number of times per second that data is transmitted.

The acceptable serial ports are COM1, COM2, COM3, and COM4. As explained in Chapter 4, the serial port can receive and transmit data one bit at a time. The signaling rate (the number of times per second that data is transmitted) is the *baud* rate. The amount of data transferred in a second is referred to as the *bps* (bits per second). With the MODE command, the baud settings adjust the amount of data sent in a fixed time. Although referred to as a baud setting, the numbers used are actually the bits per second. Baud settings are 110, 150, 300, 600, 1200, 2400, 4800, 9600, and 19200. You need to use only the first two digits of the number to set the baud rate.

Reminder:
Modems are serial devices, but they are not usually adjusted from the DOS level.

The most common peripherals which are attached to a serial port and need setting from DOS are serial printers and plotters. Although modems are serial devices, they are not usually adjusted from DOS. Communications programs, however, use the DOS functions to make adjustments to the serial ports.

With versions of DOS prior to V4, the retry feature provided two choices—no retry when a time-out error is returned, or continuous retries when the P option is included. When data is sent to the printer, the port expects to see signals from the printer indicating that the data has been received. If no signals are received within a particular period of time, a time-out error is generated. By default, DOS will not reattempt to send information to the printer and will return an error message to the screen.

Including the P option prevents the error message from being displayed, as DOS continuously retries to send the data. The Ctrl-Break key combination stops this retrying process.

With DOS V4, more retry options are available. With no retry specified, DOS does not retry to send data when a time-out error occurs.

When the retry feature is specified, you can select from three options. The B setting is equivalent to the P setting in previous DOS versions and will return a busy signal to the device driver when the port is busy. The R setting will cause a ready signal to be returned from the port even if the port is really busy. Then, when the printer does become ready, the data is ready to send and an error message will not appear on-screen. The E option is most commonly used when the printing is being done in the background (by PRINT or a network print queue). The data is not transferred to the printer until the port is not busy.

Using MODE To Set the Serial Port

To set the first serial port to communicate at 2400 bps, with 8 data bits, 1 stop bit, and no parity, you type the following:

MODE COM1 2400, N, 8, 1

or

MODE 24, N, 8, 1

With DOS V3.3 and DOS V4, you type the following to get the same settings:

MODE COM1 BAUD=24 DATA=8 STOP=1 PARITY=NONE

Note that the printer or plotter needs to be set to receive data in the same format in which the serial port is sending it—that is, the same baud rate, parity, and so on.

Using MODE To Redirect a Parallel Port to a Serial Port

The final MODE setting used with ports is the command to redirect a parallel port to a serial port.

Issuing the MODE Command To Redirect Ports

The syntax for the MODE command that changes the parallel printer to a serial printer is as follows:

MODE LPT*x*:=COM*y*:

LPT*x*: is the name of the parallel printer port; **COM*y*:** is the name of the serial port.

Understanding the Operation of MODE To Redirect Ports

Once MODE is used, DOS will channel to the serial port all I/O requests that a program sends to the parallel port. The conversion of data from byte-wide to bit-wide is done automatically by the electronics associated with the port.

Understanding the General Rules of MODE To Redirect Ports

- Any parallel port can be redirected to any serial port.
- The serial port must be initialized with its speed and data characteristics before the parallel port is redirected (see the preceding section on MODE).
- The initialization of the serial port must include the retry option if the attached device is a printer.

Reminder:
You can use MODE to redirect any parallel port to any serial port.

Using MODE To Redirect Ports

Some early programs do not directly support serial printers. To use such a serial printer, you can set a serial port and redirect a parallel port to that serial port. This process will enable you to print using a serial printer.

The serial port is initialized with DOS V3.3 using a command similar to the following:

MODE COM2 2400, E, 7, 2

With DOS V4, the command is as follows:

MODE COM2 BAUD=96 DATA=7 STOP=2 PARITY=EVEN

You can then follow the initialization command by typing the following:

MODE LPT1=COM2

All data that would normally go to LPT1 will be transmitted from COM2 at 9600 bps, with 7 data bits, 2 stop bits, and even parity.

Using MODE with DOS V4 To Change the Typematic Rate

When you press a key on the PC keyboard, the character appears on the display. If you hold the same key down for a short period of time, nothing more happens. If you continue to hold the key down, the character pressed is repeated on-screen. The number of times a second the key is repeated is known as the *typematic rate*.

With versions of DOS prior to V4, you need an add-on utility program to alter the keyboard typematic rate. You can now make this setting from DOS.

Issuing the MODE Command To Set the Typematic Rate

The syntax for the external command MODE that changes the typematic rate is as follows:

MODE CON RATE=*rate* **DELAY=***delay*

rate is the number of repetitions per second; *delay* is the time delay before DOS starts repeating a key.

Understanding the Operation of MODE To Set the Typematic Rate

Two adjustments are possible: how long you have to hold the key down before it starts repeating (the *delay*) and the number of times per second the character is generated (the *rate*).

The delay is specified in 1/4-second intervals. The range for the delay is 1 through 4, making a total possible delay of 1 second.

The rate range from 1 to 32. Table 9.4 shows the correlation between the rate value and the typematic rate.

Table 9.4
Typematic Rates

Rate	*Typematic Rate*
1	2.0
2	2.1
3	2.3
4	2.5
5	2.7
6	3.0
7	3.3
8	3.7
9	4.0
10	4.3
11	4.6
12	5.0
13	5.5
14	6.0
15	6.7
16	7.5
17	8.0
18	8.6
19	9.2
20	10.0
21	10.9
22	12.0
23	13.3
24	15.0
25	16.0
26	17.1
27	18.5
28	20.0
29	21.8
30	24.0
31	26.7
32	30.0

Notice that the increase in *typematic rate* is not linear. As the value for *rate* becomes larger, the typematic rate becomes significantly larger.

Using MODE To Set a Delay

To set the keyboard so that the delay before the key starts repeating is 0.75 seconds, and the typematic rate is 15, type the following:

```
MODE CON RATE=24 DELAY=3
```

Redirecting Input and Output to Other Devices with Redirection

As explained in Chapter 4, DOS uses three standard devices: one for input, one for output, and one for errors. Usually the console receives and accepts data. The keyboard is the standard input source. The display is the standard output and standard error device.

You type commands at the keyboard. The commands are displayed on-screen. DOS executes a command and supplies output to the screen. When an error occurs (perhaps you specified a directory command on a drive that doesn't exist), the error message is displayed on-screen.

Reminder: *Standard errors are always sent as output to the screen.*

DOS enables you to redirect the standard input and standard output information, but standard errors are always sent to the screen.

Issuing the Redirection Commands

The syntax for redirecting a program's input is as follows:

function < *inputdevice*

The syntax for redirecting a program's output is as follows:

function > *outputdevice*

The syntax for redirecting and appending to an existing file is as follows:

function >> *outputdevice*

inputdevice is the source of the input for the function. The output is sent to *outputdevice*. **function** can specify almost all DOS commands.

Any DOS output device can be used as *outputdevice*. Any DOS input device can be used as *inputdevice*.

Understanding the General Rules of Redirection

- Do not use redirection on a DOS command line which includes CALL, FOR, or IF.
- Using > and referring to an existing file causes the existing file to be overwritten with the new standard output.
- Using >> appends the standard output to an existing file, or creates the file if it does not previously exist.

Understanding the Operation of Redirection

When you use a redirection symbol, DOS either looks at a different device to receive input, or supplies the output to a different device.

Most commonly, redirected input comes from a file. Devices such as intelligent mice or bar code readers, however, can be used as the source of data. For example, a file may consist of the keystrokes used to operate a program. Alternately, the information sent from the mouse can be used as the input to a program.

Most DOS users redirect output more often than they redirect input. The **DIR > PRN** command, which redirects the output of a directory to the printer, is a common example of redirection. When you issue this command, DOS redirects the output (the listing itself) to the PRN device instead of sending it to the screen. The command produces a hard copy listing of the directory.

Another common use is redirecting output to a file. For example, to review the statistics of the **MEM/DEBUG** command, you can type the following:

MEM/DEBUG > DEBUG.MEM

Reminder: *Redirection is commonly used to redirect the output of a file to the printer or to redirect the output of one file to another file.*

When used in association with pipes and filters, discussed later in this chapter, redirection becomes even more powerful.

Using Redirection To Print the Output of Commands

To redirect to the printer a copy of the directory structure on your disk, as well as a list of the files on your hard disk, type the following:

TREE C:\ /F >PRN

The TREE command does not supply a directory listing that includes file sizes, however. You can create a full listing of all the directories on your hard disk by using redirection to append the output of the command. You specify that each directory be listed, and then specify that the output be appended to a file. You can then print the full list. Each directory command would have the following form:

```
DIR C:\NEXTPATH >> FULLLIST.DIR
```

You substitute the name of each directory in turn for C:\NEXTPATH.

If you are testing programs that require a large amount of user input, the following redirection method is useful. For example, you can type the following:

```
PROGRAM < C:TESTPAT
```

This command results in the file TESTPAT supplying the input to PROGRAM. The redirection process enables you to construct a file containing the correct keystrokes needed to operate a program. You can then test the program's basic operation before all the error-trapping sequences are included. These sequences would handle situations when the incorrect key has been typed on the keyboard.

Introducing Filters (MORE, FIND, SORT) and Piping (|)

Cue:
DOS filters can be used to modify information as it is passed from files to the screen.

DOS supplies additional functionality with its interaction with devices. DOS is responsible for channeling information between devices. It includes three elements called filters, which can be used to modify information as it is passed from files to the screen. Filters only work on ASCII text files.

The MORE filter acts like a screen buffer. Instead of information being sent directly to a device, MORE collects information in a temporary file on disk and displays it one screenful at a time. The FIND filter is used to find specific text within a file, and the SORT filter is used to sort lines of text in a file.

These filters are often used in association with the redirection symbols so that the input can come from a different source than the keyboard or be sent to a different device than the screen.

Another feature used with these commands is piping. You use the pipe symbol (|) to send output information that would normally go to the screen as input to another program. Piping is a form of redirection because it diverts information destined for a device, but it then makes the information become the input from a "device" to another program.

Buffering with the MORE Filter

The MORE filter buffers information from the standard input and sends it only to the screen, one screenful at a time.

Issuing the MORE Filter

The syntax for the MORE filter is as follows:

MORE

MORE is commonly used in two different ways:

1. **MORE** < *filename*

   filename is the input file.

2. *command* | **MORE**

   command is any command or program.

Understanding the Operation of MORE

MORE is a filter that collects information that would normally go to the screen in a temporary file on disk. When a screenful of input is obtained, MORE sends it to the standard output device all at once. The text is channeled through the MORE filter until the end of the file. Pressing any key at the `-- More --` prompt displays the next screen. Once all of the information has been viewed on-screen, the temporary file created by MORE is erased.

Cue:
The MORE filter stores in a temporary file information that would normally go to the screen; the text is then channeled through the MORE filter until the end of the file.

Understanding the General Rules of MORE

- MORE appears to do nothing when issued alone. MORE requires input to be redirected or piped.
- To view further screensful of information, press any key at the prompt `-- More --`.
- Ctrl-Break terminates the command without displaying further screens.

Using MORE To Pause the Screen

When you use MORE to pause directory listings, the filter serves a function similar to that of the /P switch available with the DIR command.

The most common use of MORE, however, is to pause the TYPE command. To read the contents of a README.DOC file one page at a time, you type the following:

TYPE README | MORE

or

MORE <README

Both methods work identically. To see the contents of a file, you can use the TYPE command. If the output of the file then flows off the screen, you can reissue the command by using the DOS editing keys. Press F3 and add the pipe character (|) and MORE.

Finding Strings of Text with FIND

Cue:
Use the FIND filter to find strings of ASCII text in files.

The FIND filter will find strings of ASCII text in files. The filter is often used in association with redirection and piping.

Issuing the FIND Filter

The syntax for the FIND Filter is as follows:

FIND */C /N /V "string" d:path\filename ...*

*"string"*specifies the ASCII characters to be searched for. *d:path\filename* is the drive, path, and file name of the file to be searched. The ellipsis (...) indicates that more than one file can be specified.

The */C* switch changes the listing to a *count* of all lines that contain *"string"*.

The */N* switch changes the listing to include line *numbers* of the lines that contain *"string"*.

The */V* switch changes the listing to include all lines not containing *"string"*.

/C and */V* can be used together. The count displayed is the number of lines not containing *"string"*.

/V and */N* can be used together. The lines not containing *"string"* are displayed with their appropriate line numbers.

Understanding the General Rules of FIND

Reminder:
Use the MORE, FIND, and SORT filters only with ASCII text files.

- Use FIND only on ASCII text files.
- The string parameter is case-sensitive. The string *LOOK* is not the same as the string *look*.
- The string is enclosed in quotes. To include a quote in a string, you type two quotes together (""). FIND will search for occurrences of ".
- The standard input device is used if *filename* is not specified.
- Wild cards are not allowed in *filename*.

Understanding the Operation of FIND

When used without options, the FIND filter reads each of the specified files and displays each line that contains a particular ASCII string. DOS takes the information that would normally go to the standard output and filters it. Lines that include the ASCII string are displayed on-screen.

If you use the */V* switch, lines that include the ASCII string are not passed on to the standard output. The */C* switch is used as a *counter*; the text itself is not passed to the screen. Use the */N* switch to locate the line *numbers* within the text file. The line numbers assigned are the line numbers in the original text file, not just sequential numbers. For example, if the third, fifth, and sixth line in the text file contains the string, the line numbers displayed are 3, 5, and 6—not 1, 2, and 3.

Like the other filter commands, FIND is often used with redirection and piping. If you need to search a text file for lines including some information, it is probably more useful for you to redirect the output to a file. Then you can use the list as a reference while you look at the whole of the original file.

Using FIND To Find Files On-Disk

The FIND command can be used to find on-disk all files that have a certain extension. To find all files that have the extension .LET, type the following:

CHKDSK C: /V | FIND ".LET"

Because the /V switch was used with CHKDSK, all files with the extension .LET are listed, including the directory the file is contained in.

The CHKDSK command with the /V option lists all the files on a disk. The output from CHKDSK that would normally go to the display is supplied as input to the FIND command. The output from FIND is displayed on-screen.

Cue:
Use the FIND filter as a word-search utility with text files.

The FIND filter can also be used with text files as a word-search utility. Consider the situation where you forget the name of the memo you sent to your boss. You know that it is either MEMO1, MEMO2, or MEMO3. You always use your boss' title *Supervisor*, in memos. Type the following:

FIND "Supervisor" MEMO1 MEMO2 MEMO3

Each line in the files containing *Supervisor* will be listed. The listing will be in the following form:

```
---------- MEMO1
Supervisor of Communications
---------- MEMO2
Supervisor of Communications
---------- MEMO3
```

Sorting Information with SORT

Cue:
The SORT filter sorts the information in an ASCII file before displaying it on-screen.

The third DOS filter is SORT. SORT takes the information from an ASCII file and sorts it before displaying it on-screen. Like FIND and MORE, SORT is often used with redirection and piping.

Issuing the SORT Filter

The syntax for the SORT filter is as follows:

SORT */R* */+n*

/R reverses the sort order.

n is the offset column for the sorting process.

Understanding the General Rules of SORT

- If *n* is not specified, column 1 is assumed.
- Lines are sorted according to their ASCII binary values with two exceptions:
  1. SORT is not case-sensitive. For example, A and a are sorted as they occur in the source file.
  2. Characters with values over 127 are sorted in the order determined by the current COUNTRY code setting. (See Chapter 15 for more information.)

- The output file name must be different from the input file name when redirection is used.

Understanding the Operation of SORT

The SORT filter filters information that would normally go to the standard output device. The text from the input is analyzed on a line-by-line basis and sorted according to the ASCII binary values of the characters. This order is alphabetical, but no discrimination is made between upper- and lowercase letters.

When used with redirection or piping, sophisticated sorting can occur. The most common use of SORT is to sort a directory listing into a text file, which can then be printed.

The offset option in the command indicates the leftmost column to be sorted. With a directory listing, you can sort by date or file size rather than just by root name. Table 9.5 lists the offset values for a directory listing.

Table 9.5
Offset Values for SORT

Offset value	*Sorting by*
1	Root name
10	File extension
14	File size
24	File date
34	File time

Note that the sorted output is a file or text on-screen. This is not the same as altering the directory listing itself. The system directory is unchanged.

Using SORT To Sort Subdirectories

Redirection and piping, as well as the FIND and SORT filters, can be used in a variety of ways. In DOS V4, for example, you can create a sorted list of the subdirectories in the root directory by using the process outlined in the next paragraphs.

Issue the following command:

TREE C:\ /A | FIND "---" | SORT /+2 >TEMP.LST

The TREE command with the */A* switch produces a listing that includes the *alternate* graphics character set. The FIND filter then removes all lines that are

not subdirectory names. The output from FIND is sorted by SORT on column 2, and the output is redirected to a temporary file.

Next, you use the temporary file as the input to the FIND filter, which removes all directories that are not in the root.

```
FIND /V "|" <TEMP.LST
```

The result is an alphabetical list of all subdirectories in the root directory.

Summary

This chapter discusses the commands associated with changing the way DOS interacts with devices. These range from basic commands that clear a screen to advanced filtering and redirection. The following important points were covered:

- Use the CLS command to clear the currently active display.
- You use the GRAPHICS command to print graphics screens on supported printers.
- The PRINT command is a background printing program.
- The ASSIGN, JOIN, and SUBST commands are used to change the device names assigned to disk drives and directories.
- Use the KEYB command to change keyboard drivers.
- Use the CTTY command to redirect standard input and output devices.
- Use the options available with the MODE command to adjust settings for serial and parallel ports.
- You can use MODE with DOS V4 to alter the keyboard typematic rate.
- The MORE, FIND, and SORT filters are used with text files. The filters are frequently used with redirection and piping.

The next chapter, which begins Part III of this book, introduces the DOS-provided line editor, the EDLIN program.

Part III

Expanding Your Use of DOS

Using EDLIN
Understanding Batch File Basics
Understanding ANSI.SYS
Learning Advanced Batch File Techniques

10

Using EDLIN

EDLIN, the mini text editor, is DOS's forgotten child. Because EDLIN is provided with every copy of DOS, the text editor has easily "outsold" Microsoft Word, WordPerfect, WordStar, or any other word processing program. Yet only a relatively small number of DOS users learn to use this simple, flexible program. As you will discover in this chapter, this neglect is undeserved.

Text editors, which actually predate word processing programs by a number of years, were originally developed for computer programmers who used them to enter and edit lists of commands for the computer. Frequently, these commands were programs eventually translated (or compiled) into the machine language understood by the computer. Programmers liked the ability to edit lines, add new lines, or move existing lines to other parts of the program.

EDLIN isn't limited to programming, however. You can use it as a handy tool to write short notes, "to-do" lists, or memos. You can use EDLIN to make changes in special configuration files used by DOS and many applications programs. The DOS CONFIG.SYS file, the Microsoft Windows WIN.INI file, and similar files for other programs are simple ASCII files you can edit with EDLIN. In Chapter 11, you will learn that EDLIN is useful for typing or editing a series of commands in special batch files.

If you understand its programming-environment heritage, you can see that EDLIN makes a lot of sense. It works with a single line at a time; programmers write or edit commands as a series of separate lines. EDLIN provides a line number before each line; programmers use such labels to keep track of individual program lines. EDLIN enables you to move a line or a group of lines to another location in a file; programmers need these features. The EDLIN features also are handy for other text-editing chores.

Understanding EDLIN is helpful even if you use another word processing program to create ASCII files (for more information on ASCII files, see Chapters 3 and 7). You can load EDLIN, type a few lines, and be finished in about the same time required to load a full-featured word processor. Word processors may require special syntax or formatting to produce ASCII files; EDLIN generates

nothing but plain ASCII text. Additionally, if you find yourself using someone else's computer, your favorite text editor may not be available. EDLIN nearly always is close at hand.

This chapter explains how to use EDLIN to create and edit a simple memo. You'll be pleased to know you can gain EDLIN proficiency in 30 minutes or less. This chapter also contains information on additional, more advanced EDLIN commands.

Starting EDLIN

Although EDLIN is furnished with DOS, it is a separate program. The actual program is found on one of the diskettes supplied with DOS. If you use a computer that does not contain a hard disk drive, you may have to copy EDLIN from the original DOS diskettes (or a backup) to a diskette with enough room to store a few ASCII files. The DOS disk on which EDLIN is located varies with the DOS version and disk format. With DOS V4 and 3 1/2-inch disks, for example, you can find EDLIN on the disk labeled *DOS Operating*. Some dealers include the EDLIN documentation in the DOS package; others feature the documentation as a separate item.

If you have a hard disk, you probably copied EDLIN to the directory (often named C:\DOS) in which you store other DOS programs such as XCOPY.EXE. If not, take the time to copy EDLIN to that directory now.

Cue:
To run EDLIN, type ***EDLIN*** *followed by the name of the file to be created or edited.*

To run EDLIN, you simply type **EDLIN**, followed by the name of the file you want to create or edit. EDLIN itself must be located in the current directory or one of the directories listed in your DOS path (the PATH command is explained in Chapters 3 and 7). If EDLIN is stored in the same directory as your other DOS files, you probably already have defined a path that allows DOS to locate it with no trouble. If the EDLIN directory is not specified in the PATH command, you can load EDLIN by changing to the directory where EDLIN resides or (if you have DOS V3.0 or later) by placing the full path name before the EDLIN commands.

Creating a Memo with EDLIN

In this section, you will learn to use EDLIN to create a simple memo. Type the following command at the DOS prompt to start EDLIN with the file MEMO.TXT:

```
C>EDLIN MEMO.TXT
```

If EDLIN is located in another directory not indicated by the path, you can include the path name in the command:

```
C>C:\UTILS\EDLIN MEMO.TXT
```

Either of these commands tell EDLIN to begin working with a file called MEMO.TXT in the current directory (even if EDLIN itself is located in another directory). If MEMO.TXT already exists and is smaller than about 64,000 bytes (64K), EDLIN loads it all into memory. If the file is larger than 64K, only the first 64K of the file is loaded.

For this exercise, you will create a new file called MEMO.TXT. (Actually, the file isn't created until the end of the editing session. Until then, EDLIN works with an intermediate file that has the extension .$$$, regardless of what you specify when you start EDLIN. The reason for this is explained later in the chapter.) If you want to work with a file in another directory, place the path name in front of the file name, as in the following example:

```
C>C:\DOS\EDLIN C:\MEMOS\MEMO.TXT
```

This command loads EDLIN (no matter which directory is the current one) and begins work with a file called MEMO.TXT in the directory C:\MEMOS.

Once EDLIN is started, you will see the EDLIN prompt. Although the default DOS prompt is the drive name and greater-than symbol (such as A>, C>, and so on), EDLIN's prompt is a simple asterisk:

Reminder:
The EDLIN prompt is an asterisk (*); the DOS prompt is the drive name and the greater-than (>) sign.

```
*
```

When you see this prompt, EDLIN is waiting for your instructions. Each EDLIN command consists of a letter (either upper or lowercase), which can be preceded or followed by numbers that tell EDLIN the number of lines you want to work with or the line numbers you are referring to. Some commands also enable you to specify strings of characters to search for or replace.

Inserting Text

Because the MEMO.TXT document you created in the preceding section is empty, you first need to enter some text. **I** (Insert Lines) is EDLIN's command for inserting text (even when the file is empty, text is inserted). To tell EDLIN that you want to insert text into the current file, you type the following at EDLIN's prompt:

```
*I
```

EDLIN responds with the following:

```
1:*
```

Notice that the colon is preceded by a line number, which tells you that EDLIN is currently in Insert Lines mode. If a line number, a colon, and an asterisk are the only things shown on a line, EDLIN is waiting for you to enter text. Now type the first line of the memo:

```
1:*MEMO TO: Scott N. Hollerith, President
```

Press Enter at the end of the line. EDLIN responds with the following:

```
2:*
```

Cue:
*When shown with a line number, the asterisk * marks the current line.*

Don't confuse the asterisk shown after the line number with EDLIN's prompt. When shown with a line number, the asterisk marks the *current line*. The current line is the line currently being entered or, when you are editing lines, the line most recently edited. You will see the importance of the current line later in this chapter.

EDLIN is now ready for you to enter a second line. Type the rest of the following memo, pressing Enter at the end of each line:

```
2:* From: Otto Wirk, Research Chief
3:* Subject: New Dual Processor Micro
4:* Date: Sunday
5:*
6:* Our testing confirms that we should indeed
7:* include two microprocessors in the next model
8:* TLS-8E. This increases the odds that one will
9:* work. If both happen to function, they can
10:* take turns
```

Cue:
To stop inserting text, press Ctrl-C or Ctrl-Break.

To leave line 5 blank, simply press Enter without typing anything. When you are finished with the memo (for the moment, anyway), stop inserting text at line 10 by pressing Enter to finish the line. Then press Ctrl-C or Ctrl-Break (hold down the Ctrl key and press C or hold down the Ctrl key and press Break). The EDLIN asterisk prompt returns.

Even though you haven't saved the memo to disk yet, suppose that you forgot to include several other staffers on the distribution list. The **I** command lets you fix this error quickly.

Type the following and press Enter:

```
*5I
```

Cue:
*Type a line number in front of the **I** (Insert Lines) command to insert lines before the specified line number.*

EDLIN responds with the following:

```
5:*
```

If you type a number in front of the **I** command, EDLIN enables you to enter text beginning with a new line preceding the line you specify. In this example, EDLIN inserts a *new* line 5 in front of the *old* line 5. EDLIN renumbers the lines as necessary to accommodate the new line or lines. The line numbers aren't

part of the memo you are creating; EDLIN uses line numbers only to help you keep track of the various lines currently in memory. The line numbers are not included in the file you will store to disk later.

At the **I** (Insert Lines) prompt, type this line:

```
5:* CC: Bill Kem, Marketing; Betty Site, Testing
```

Press Enter at the end of the line. A new blank line 6 appears. Because you don't want to add another line to the file, press Ctrl-C or Ctrl-Break to stop inserting lines.

Listing Text

If you want to see what the memo looks like with the addition, you can use EDLIN's **L** (List Lines) command to display the lines currently in memory. You can use the **L** command to look at a single line or a range of lines. Up to 23 lines can be shown on-screen at one time. Because the sample memo is shorter than 23 lines, you don't have to type in a range of line numbers to be displayed. Instead, specify only the first line you want displayed. Type the following:

Reminder:
*Type only the first line number in front of the **L** (List Lines) command if the file is less than 23 lines long.*

```
C*1L
```

EDLIN responds by showing you the entire file so far:

```
 1: MEMO TO: Scott N. Hollerith, President
 2: From: Otto Wirk, Research Chief
 3: Subject: New Dual Processor Micro
 4: Date: Sunday
 5: CC: Bill Kem, Marketing; Betty Site, Testing
 6:
 7: Our testing confirms that we should indeed
 8: include two microprocessors in the next model
 9: TLS-8E. This increases the odds that one will
10: work. If both happen to function, they can
11: take turns
```

Notice that the original blank line 5 has become line 6. In addition, the asterisks that followed each line number when the text was entered are gone. They, like the line numbers, appear on-screen only to guide you.

For practice, list only a few lines at a time using the **L** command. Type the following:

Reminder:
When listing lines, use a comma to separate the beginning and ending line numbers.

```
*1,5L
```

EDLIN displays the first five lines of the memo: lines 1 to 5. Notice that when you type a line range, you separate the numbers with a comma instead of a hyphen, space, or other character.

You can list a single line by typing that line as the range. For example, you can type the following:

```
*1,1L
```

Editing a Line

If you listed the entire memo, you may have noticed that no period was typed at the end of the last sentence. This error can easily be fixed.

At EDLIN's * prompt, type the following:

```
*11
```

Notice that you do not type a specific command in this instance; you type just a line number at the EDLIN prompt to indicate that you want to edit line 11 of the current file. EDLIN responds by displaying the text of the specified line with the line number, colon, and asterisk prompt on the following line:

```
11:* take turns
11:*
```

Reminder:
Press F3 to copy the text from the specified line to the line being edited.

No text is displayed on the second line. If you want, you can type a new line to replace the old one. Because you only want to make a small change at the end of the line, however, press F3. EDLIN displays the entire line:

```
11:* take turns
11:* take turns
```

Press the period key (.); then press Enter. EDLIN displays line 12 just as it did line 11. Press Ctrl-C or Ctrl-Break to stop editing.

The same function keys used as DOS editing keys are used with EDLIN. Recall that F3 is used to redisplay an entire command line. In the example you just worked through, F3 enables you to begin editing at the end of the specified line. If you want to replace the final word in the line, you can press Backspace four times and then type a new word. Later in this chapter, you will see how the other DOS editing keys are used with EDLIN.

For now, type **1L** at the EDLIN * prompt to list the memo. It looks much like it did before, with two changes:

```
1: MEMO TO: Scott N. Hollerith, President
2: From: Otto Wirk, Research Chief
3: Subject: New Dual Processor Micro
4: Date: Sunday
5: CC: Bill Kem, Marketing; Betty Site, Testing
6:
7: Our testing confirms that we should indeed
```

```
 8: include two microprocessors in the next model
 9: TLS-8E. This increases the odds that one will
10: work. If both happen to function, they can
11:*take turns.
```

The period you typed at the end of line 11 is added, and line 11 now has an asterisk following the colon. The asterisk indicates that line 11 has been marked by EDLIN as the current line. Several EDLIN functions use the current line as a default if you don't enter a new value when you type the command. Use the **L** (List Lines) command and look for the asterisk after the line number to discover which line is the current line.

Cue:
To determine the current line, look for the asterisk after the line number in a listed file.

Saving Text

Before you modify the memo further, save it to disk so that you can retrieve it in this finished form later. To save the current file, you only need to end the EDLIN session with EDLIN's **E** (End Edit) command. The DOS prompt then returns. EDLIN automatically writes the file to the disk with the changes (if any) you made in the current session.

Reminder:
*Use the **E** (End Edit) command to return to DOS and save the current EDLIN file.*

Try this technique now for practice. At the EDLIN prompt, type the following:

```
*E
```

Remember to press Enter to activate the EDLIN command. The DOS prompt reappears. Notice that you don't get any "feedback," such as a message like "saving file to disk." No such message is really needed: when you exit EDLIN with the **E** command, the text is always stored. To abandon changes, use EDLIN's **Q** (Quit Edit) command to end the EDLIN session. The **Q** (Quit Edit) command is discussed in a later section of this chapter.

Reminder:
Remember to press Enter to activate all EDLIN commands.

Printing the Memo

While you are at the DOS prompt, you can practice printing the memo. EDLIN does not have a specific print command, but to print an EDLIN file, you can use one of several DOS commands: PRINT, COPY, and TYPE.

Cue:
To print an EDLIN file, you can use one of several DOS commands: PRINT, TYPE, or COPY.

You can use the PRINT command to copy the file to the printer. (The PRINT command is discussed in more detail in Chapter 9.) Alternatively, you can use the COPY command to copy the file to the printer. The TYPE command also can be used to redirect the file to the printer. Redirection is discussed in Chapter 4, and TYPE and COPY are discussed in Chapter 7.

To print your memo, type one of the following commands at the DOS prompt:

```
C>PRINT MEMO.TXT
```

```
C>TYPE MEMO.TXT>PRN
```

```
C>COPY MEMO.TXT PRN
```

If you have not used the PRINT command or directed output to the PRN device since the last time you rebooted, DOS may ask you to specify the print device and offer PRN as the default. If so, press Enter to confirm the default choice and start printing. If you have used DOS to print a file since you rebooted, PRINT is already resident in memory and you don't need to specify the print device.

Because you still have some work to do with MEMO.TXT, reload the file into EDLIN. To start EDLIN with an existing file, type the following:

```
C>EDLIN MEMO.TXT
```

The prompt reappears. Typing **L** after you load a file is good practice so that you can list the file and confirm that EDLIN has indeed loaded the text you intend to edit.

Deleting Lines

The last of EDLIN's basic commands is **D** (Delete Lines). As with the **L** (List Lines) command, you can work with a single line or a range of lines. The syntax is the same. Delete lines 5 and 6 of the memo by typing the following line at the EDLIN prompt:

```
*5,6D
```

List the lines of the memo (type **1,11L**) to see what has happened. The memo now looks like this:

```
1: MEMO TO: Scott N. Hollerith, President
2: From: Otto Wirk, Research Chief
3: Subject: New Dual Processor Micro
4: Date: Sunday
5:*Our testing confirms that we should indeed
6: include two microprocessors in the next model
7: TLS-8E. This increases the odds that one will
8: work. If both happen to function, they can
9: take turns.
```

EDLIN again has renumbered the lines to accommodate the changes. Line 5, the line marked by the asterisk, is the current line because it is the first line that follows the lines you removed.

You can end this practice EDLIN session by typing **Q**. EDLIN asks whether you want to abort the session. Respond by pressing Y. The changes you made (deleting two lines) are not saved. The original version of MEMO.TXT remains on disk.

Learning More about EDLIN's Features

In the process of creating a memo with EDLIN, you learned some of the program's basic features. EDLIN also includes some additional editing features and commands. These are discussed in the remaining sections of this chapter.

Using EDLIN's /B Switch

EDLIN has an additional parameter, the */B* switch, that can be used when you start the program. You add the */B* switch to the initial EDLIN command at the DOS prompt, as in the following example:

```
C>EDLIN MEMO.TXT /B
```

The */B* switch tells EDLIN to read the entire file, based on its size as listed in the disk directory.

EDLIN normally reads a file only until it encounters the first end-of-file (EOF) character in the file. The EOF character is the Ctrl-Z (^Z) traditionally used to mark the end of ASCII files. Most of the time, you want EDLIN to read to the EOF character. Sometimes, however, you may want to edit a file that contains the EOF character within the file itself.

Cue:
Use the /B switch if you have end-of-file characters in your EDLIN file, or EDLIN will stop reading the file at the first end-of-file character.

In its normal mode, EDLIN loads only the first part of such a file. By using the /B switch, you tell EDLIN to load the entire file, ignoring any Ctrl-Z characters it encounters in the file. You also can load program files (files that have the extension .COM or .EXE). Because program files are also called *binary* files, the switch is called /B. You have little reason to load such binary files; other tools like DEBUG are better suited for editing true binary files.

Understanding How EDLIN Works with Files

Remember that when you start EDLIN, it tries to find the file you specify in the current directory (or in the specified path if you precede the file name with a path name). If the file is not found, EDLIN creates the file with the extension .$$$, regardless of the extension you assigned. If you type the following, for example, EDLIN attempts to load a file called MEMO.TXT from the directory C:\MEMOS:

Reminder:
EDLIN creates a new file and appends the extension .$$$ if the file you specify does not exist.

```
C>EDLIN C:\MEMOS\MEMO.TXT
```

If EDLIN cannot find such a file, it creates a new file called `MEMO.$$$`.

Reminder:
When you save an EDLIN file with the ***E*** *(End Edit) command, the* `.$$$` *extension is replaced with the extension you specified.*

If a new file is created, the message `New file` appears. When you save the file by ending the session with the **E** (End Edit) command, the file is renamed from the .$$$ extension to the .TXT extension specified when you started EDLIN.

If the C:\MEMOS\MEMO.TXT file specified when you started EDLIN is found, EDLIN copies the original file and gives it a .$$$ extension. If the file is less than 64K in size, EDLIN loads the file and displays the following message:

```
End of input file
```

When you save the file, the original MEMO.TXT file is renamed with the extension .BAK; the temporary .$$$ file is given the original MEMO.TXT file name and becomes a permanent disk file. If you decide to quit the session without saving the file (using the **Q** (Quit Edit) command), the last version of the file on disk is retained unchanged (if you were editing an existing file). If you were working with a new file when you used the **Q** (Quit Edit) command, the temporary .$$$ file is erased.

These options and conditional statements may sound confusing, but they all boil down to one thing: EDLIN enables you to quit editing at any time without destroying the original disk version of your file.

EDLIN can work with only a limited amount of information at one time. Each line of text, for example, can have up to 253 characters, and each file can have from 1 to 65,529 line numbers, depending on file size. If the file is longer than 65,529 lines, EDLIN works with the file in sections. You can load the first section of the file into memory, work with it, and then save it to disk. EDLIN then enables you to load more of the file and work with that section. The process can be repeated until editing is complete. This section-by-section editing process is explained in the next section.

Learning Special Editing Techniques

In this chapter, you learned to edit a line by typing the line number at the EDLIN asterisk prompt. If you press Enter without typing a line number, the current line, indicated with the **L** (List Lines) command by an asterisk, is edited.

Cue:
You can cancel changes to the current line by pressing Esc or by pressing Enter when the cursor is in the first position on the line.

When editing a line, you can use a combination of keys to edit and change the text line. You can add information by typing up to 253 characters on the line (EDLIN wraps the line automatically at the right edge of the screen). After you have edited the line, press Enter. The cursor returns to the * prompt.

If you change your mind or make a mistake as you are editing, press Esc. This action cancels the changes to the current line. You also can cancel changes to the current line by pressing Enter when the cursor is in the first position on the current line.

You can also add special control characters such as Ctrl-Z, the end-of-file character, to your text files. To enter a control character, press Ctrl-V and then type the desired character in uppercase letters. To type a Ctrl-Z character, for example, press Ctrl-V and Z.

Another character you may want to enter is the Escape character (ASCII character #27). Batch files (discussed in Chapter 11), make use of the Escape character to send special codes to ANSI.SYS (discussed in Chapter 12). If you press the Esc key to try to enter the Escape character, a backslash (\) appears on the line and the cursor moves to the next line, abandoning what you just typed. To enter the Escape character in a file, first press Ctrl-V and then press the left square bracket ([).

With EDLIN, you also can add graphics characters, such as those used to make a box. To add graphics characters, however, you must make sure that ANSI.SYS is installed as a device in CONFIG.SYS. You learn more about ANSI.SYS and CONFIG.SYS later in this book.

To enter a graphics character, hold the Alt key and use the numeric keypad to type the ASCII code for the character you want to insert. (A complete list of ASCII codes is found in Appendix B of this book.) To enter a vertical bar, for example, hold the Alt key and type **179** on the numeric keypad. When you release the Alt key, a vertical bar appears on-screen.

Using the Editing Keys

You can use the function keys F1 through F6 with EDLIN to reenter and correct text lines. The function keys available for editing in EDLIN are explained in table 10.1.

Table 10.1
EDLIN Key Combinations

Key	*Function*
F1 or right arrow	Copies one character from the edited line to the current line.
F2 *char*	Copies all the characters from the edited line to the first occurrence of the character *char*.
F3	Copies all the remaining characters from the edited line to the current line. By pressing F3 without first typing anything on the current line, all characters from the edited line are copied to the current line. If you have already started to edit the current line, pressing F3 copies the remaining characters.

Table 10.1—*continued*

Key	*Function*
F4 *char*	Copies all the characters from the edited line found after the value of the character *char*.
F5	Copies all the characters typed on the current line into the keyboard buffer. When you press F3, the copied characters transfer to the edited line.
F6	Creates a Ctrl-Z, (^Z) end-of-file, character.
F7	Creates a null, Ctrl-@.
Ins	Acts as a toggle switch between insert and overtype modes. After you press Ins, you can insert characters at the current cursor position. Press Ins again to turn off insert capabilities.
Del	Erases one character to the right of the current cursor position.
Esc	Causes the current line to be ignored.

Using the EDLIN Commands

Table 10.2 is a complete list of the commands available with EDLIN. Each command is listed with its function and any available parameters. The rest of this chapter describes each of these commands and their parameters in more detail.

Table 10.2
EDLIN Commands

Command	*Syntax*	*Function*
A (Append Lines)	*n***A**	Loads the next section of a file into memory if the file is too large to fit into memory
C (Copy Lines)	*sline*,*eline*, **dline**,*count***C**	Copies one or more lines within a file
D (Delete Lines)	*sline*,*eline*,**D**	Deletes one or more lines from a file
E (End Edit)	**E**	Ends EDLIN and saves the edited file
I (Insert Lines)	*n***I**	Inserts new lines of text into the file

Command	*Syntax*	*Function*
L (List Lines)	*sline*,*eline***L**	Lists the file's contents on-screen; does not change the current file
M (Move Lines)	*sline*,*eline*,**dlineM**	Moves and relocates lines within a file
P (Page)	*sline*,*eline***P**	Lists the file's contents *and* changes the current line
Q (Quit Edit)	**Q**	Quits without saving changes to the file
R (Replace Text)	*sline*,*eline?* **Rstext**<**F6**> **rtext**<**D**>	Replaces text within a file
S (Search Text)	*sline*,*eline?***Sstext**	Searches for a specified text string
T (Transfer Lines)	*n***T***d:***filename.ext**	Transfers the contents of one file into another file
W (Write Lines)	**W**	Writes a specified number of lines to disk from memory

The following definitions are assumed in table 10.2.

<**F6**>	The F6 function key is pressed.
count	The number of times the procedure is to be repeated.
d:	Drive name. If unspecified, the default drive is used.
dline	*Destination* line number. The text to be moved or copied is placed *in front of* the line number specified by *dline*.
eline	*Ending* line number. If unspecified, the operation usually affects only the *sline* (starting line).
filename.ext	A legal DOS file name.
n	An integer representing a line *number* or a total *number* of specified lines.
rtext	Series of alphanumeric characters to *replace stext*.
sline	*Starting line* number in a group of lines. If unspecified, the operation usually assumes the current or default line.
stext	Series of alphanumeric characters for which to *search*.

A—The Append Lines Command

EDLIN can hold in memory only a certain number of lines at one time. To work with larger files, you load a section of a file into memory, make your editing changes, write that section to disk, and then load the next section. The **A** (Append Lines) command is one of the tools you use to perform this process.

*Cue: Use the **A** (Append Lines) command to load a portion of a large file into memory.*

In actual practice, you rarely use EDLIN in this manner, because the program is somewhat clumsy for really long files. A full-featured word processor is a better choice. As a safeguard against the day when you may want or need to use EDLIN with a long file, however, you should learn to use the **A** and **W** (Write Lines) commands. They are really quite simple.

The **A** command adds the number of lines you specify from the disk and places them into memory. Lines are appended from the disk to the end of the lines currently in memory. If memory can hold no more lines, you may first need to write some of the lines already resident to disk. Otherwise, the first lines (equal to the number you have appended) will be dropped. You will learn in the next section how to write lines to disk with the **W** command.

The syntax for the **A** (Append Lines) command is as follows:

*n***A**

n is the number of lines to place into memory.

If you see the message `End of input file` when you start EDLIN, you don't need the **A** command. You use **A** only when the file being edited is too large to fit in memory.

To append lines, type a number representing the total number of lines you want appended to the end of memory. Then type **A** and press Enter. The cursor returns to the `*` prompt.

Continue the process of writing and appending lines until you see the following message:

```
End of input file
```

After this message appears, all lines have been read into memory. If you don't use the **W** (Write Lines) command before you use **A**, you may lose previously edited corrections.

If you do not specify the number of lines to be appended, any remaining lines in the file are appended to the file in memory until available memory is 75 percent full. The number of lines will vary depending on the length of the individual lines.

W—The Write Lines Command

Use the **W** (Write Lines) command to write a specified number of lines in memory to the disk. The syntax for the **W** command is as follows:

*n***W**

n is the number of lines to write to the disk.

You don't need this command if you see the message `End of input file` when you start EDLIN. Use this command only when the file being edited is too large to fit into memory. Use this command before appending new lines to the file being edited. When you write, the lines are removed from memory, thus leaving room for you to append.

Cue:
*Use the **W** (Write Lines) command when the file being edited is too large to fit into memory.*

To use this command, type the number of lines you want to write. Type **W** and press Enter; the cursor returns to the * prompt. If you want, you can continue appending lines for editing. Continue with this process until you see the following message:

```
End of input file
```

After this message appears, all lines have been read into memory; using the **W** (Write Lines) command is no longer necessary because you have written the previous lines in the file to the disk, and all the remaining lines can fit into memory. Save the remaining lines in memory by ending the editing session with the **E** (End Edit) command.

If you don't specify the number of lines to be written, the remaining lines, starting from line 1, are saved to disk until available memory is 25 percent full.

C—The Copy Lines Command

You can copy one or more lines from the current file to the line number specified with the **C** (Copy Lines) command. The syntax of this command is as follows:

sline,*eline*,**dline**,*count***C**

sline is the starting line, and *eline* is the ending line to copy from. **dline** is the line in front of which the copied lines are to be placed, and *count* is the number of times to copy the text.

To copy lines, type the starting line number where the copying is to begin. Press the comma (,) key and type the ending line number. You have just indicated the range of lines to copy. Press the comma again and type the line in front of which the copied lines are to be inserted. If you want more than one copy of the range, type the comma and the number of copies you want. Finish the command by typing C and pressing Enter. The cursor returns to the *

Reminder:
When you copy a range of lines, the copied lines are inserted in front of the specified line number.

prompt. Use the **L** (List Lines) command to check the insertion of the copied lines. The copied lines appear in front of the specified **dline** number.

Try the following example. Start EDLIN and load MEMO.TXT (the file you created in a previous section). Then type **L** at the EDLIN prompt to see the following listing:

```
1:*MEMO TO: Scott N. Hollerith, President
2: From: Otto Wirk, Research Chief
3: Subject: New Dual Processor Micro
4: Date: Sunday
5: CC: Bill Kem, Marketing; Betty Site, Testing
6:
7: Our testing confirms that we should indeed
8: include two microprocessors in the next model
9: TLS-8E. This increases the odds that one will
10: work. If both happen to function, they can
11: take turns.
```

Now, type the **C** (Copy Lines) command as follows:

```
*2,4,6C
```

In this syntax, 2 is the starting line, 4 is the ending line, and 6 is the destination line.

Type **L** again to see what happens:

```
 1: MEMO TO: Scott N. Hollerith, President
 2: From: Otto Wirk, Research Chief
 3: Subject: New Dual Processor Micro
 4: Date: Sunday
 5: CC: Bill Kem, Marketing; Betty Site, Testing
 6:
 7: Subject: New Dual Processor Micro
 8: Date: Sunday
 9:
10: Our testing confirms that we should indeed
11: include two microprocessors in the next model
12: TLS-8E. This increases the odds that one will
13: work. If both happen to function, they can
14: take turns.
```

EDLIN has copied lines 2 through 4 of the original memo and placed them before line 6. EDLIN then renumbered the lines from the original line 6 (which becomes line 9) on.

Cue:
Include a count value in the ***C*** *(Copy Lines) command to repeat the copied lines.*

You can repeat the number of times a range copy is repeated. For example, the command **3,3,9,7C** copies the contents of line 3 beginning at line 9, inserts the

copy in front of line 9, and repeats the copy seven times. The old line 9 is renumbered as line 16.

If the starting or ending line number is omitted from the syntax of the **C** (Copy Lines) command, the current line is used as the missing parameter. If the current line is line 3, for example, and you type **3,,9C**, only line 3 is copied to line 9. If you type **,5,9C**, lines 3 through 5 are copied in front of line 9.

When you copy lines, do not overlap line numbers. Typing **2,7,5C**, for example, results in an error: lines 2 through 7 overlap line number 5.

Caution:
Do not overlap line numbers when you copy lines.

D—The Delete Lines Command

Use the **D** (Delete Lines) command to remove one or more lines from the file being edited. You may want to use the **L** (List Lines) command to note the current line numbers before you delete. Listing lines serves as a double-check against losing important lines in your file.

The syntax of the **D** (Delete Lines) command is as follows:

sline,*eline***D**

sline is the starting line number, and *eline* is the ending line number of the range to delete.

To delete just one line, type the line number to be deleted and then type **D**. Then press Enter. To delete line number 2, for example, type **2D** and press Enter.

To delete a range of lines, type the starting line number, a comma, and the ending line number of the range of lines to be deleted. Complete the command by typing **D** and pressing Enter. The cursor returns to the * prompt.

I—The Insert Lines Command

As you recall, the **I** command adds lines of text to a newly created file or inserts new lines of text between existing lines in an existing file or at its end. A few features of this command, however, weren't explained earlier in this chapter. The syntax for the **I** (Insert Lines) command is as follows:

Cue:
*The **I** (Insert Lines) command inserts lines into a newly created or existing file.*

*n***I**

n is the line on which insertion is to begin.

To insert lines with EDLIN, type the line number at which you want to begin inserting lines; then type **I**. If a line number is not specified, insertion begins at the current line. If you are creating a new file, simply type **I** and press Enter; new files always begin insertion at line 1.

If you do not specify a line number, the insertion takes place before the current line. The **I** command renumbers the current line and the lines that follow to reflect the new lines added.

Cue:
Use the pound symbol (#) to insert lines after the last line number.

To insert lines at the end of the file, use the pound symbol in place of the line number (**#I**). Insertion begins after the last line number and continues until you press Ctrl-C.

L—The List Lines Command

You use the **L** (List Lines) command frequently. As you learned when you were writing MEMO.TXT, the syntax for the **L** command is as follows:

sline,*eline***L**

sline is the starting line, and *eline* is the ending line of the range to be listed.

Reminder:
Minimize changes to line numbers by making changes to the end of the file first.

You can minimize changes in line numbers if you make changes to your file by working backwards. Make the changes first to the end of the file and proceed backwards to the beginning of the file. This method leaves the early line numbers unchanged, so that you do not have to list the file after every insertion to determine the line numbers of the rest of the file.

You can use the **L** command for more than listing the entire file. To list and display a range of lines, first type the starting line number where the display is to begin. EDLIN will display 23 lines starting with the line you specify; then type **L** and press Enter. You can list a block of lines by typing the starting line number, pressing the comma (,), typing the ending line number, typing **L**, and pressing Enter.

Reminder:
*The **L** (List Lines) command does not change the current line number.*

The **L** command works like EDLIN's **P** (Page) command with one exception: **L** does not change the line EDLIN thinks of as the current line; **P** *does* change the current line number.

If you omit the starting line number, **L** starts displaying the 11 lines preceding the current line. The listing continues until the ending line number is reached. If the current line were 23, for example, and you typed **,40L** and pressed Enter, lines 12 through 30 would appear on-screen.

If you omit the ending line number, 23 lines are displayed, beginning with the starting line. If the current line were 23, for example, and you typed **12L** and pressed Enter, the lines 12 through 30 would appear on-screen.

Cue:
Use + and − symbols to refer to lines around the current line.

You can reference line numbers in relation to the current line by using a plus (+) or minus (−) sign. Type **−5L** to list the five lines preceding the current line. To list the nine lines after the current line, type **+9L**.

If you cannot remember or find the current line number, use a period (.) to specify the current line number. To list the current line number, for example, type **.L** and press Enter. EDLIN interprets the period to mean the current line number.

Cue:
*Use the **.L** command to list the current line number.*

Use the **L** (List Lines) command frequently with EDLIN. The **L** command can prevent you from making a mistake when using commands that affect line numbering.

P—The Page Command

Use the **P** (Page) command to list the specified block of lines on a full 23-line display. The **P** command is similar to the **L** (List Lines) command except that the current line changes each time you use **P**. Use the **P** command to move through the file being edited one page (23 lines) at a time. The syntax of the **P** command is as follows:

sline,*eline***P**

sline is the starting line, and *eline* is the ending line of the range to be displayed.

When listing lines by page, type the line number where the block to be displayed is to begin. Then type a comma (,) followed by the ending line number of the block. If the ending line number is omitted, the 23 lines from the starting line number are displayed. Finish the command by typing **P** and pressing Enter.

The **P** command works like the **L** command, except that **P** changes the current line. Use the Pause key or press Ctrl-S to stop the display from scrolling when you display more lines than will fit. When you use **P**, the last line displayed becomes the new current line.

Reminder:
*Using the **P** (Page) command changes the current line number: the last line displayed becomes the new current line.*

If you omit the starting line, the **P** command begins the display after the current line. The listing, displaying 23 lines at a time, continues until the specified ending line is reached. If you omit the ending-line parameter, 23 lines are displayed from the starting line or the current line. If you type **12P** and press Enter, lines 12 through 35 appear on-screen. If you type only **P**, the 23 lines after the current line appear. Type **12,20P** to display lines 12 through 20. If fewer than 23 lines are in the file after the current line, typing **P** displays the lines until the lines run out.

Consider using the **P** (Page) command frequently. It can prevent you from making a mistake when you use commands that affect line numbering.

M—The Move Lines Command

The **M** (Move Lines) command moves one or more lines within a file. The **M** command is similar to the **C** (Copy Lines) command. The syntax for the **M** command is as follows:

sline,*eline*,**dlineM**

sline is the starting line, and *eline* is the ending line in the range to be moved. **dline** is the destination line in front of which you want to insert the moved lines.

To move lines of text, type the starting line number of the range of lines to be moved. Press the comma (,) and then type the line number that ends the range of lines to be moved. Then type a comma and the line number in front of which you want to move the specified range. Complete the command by typing **M** and pressing Enter.

Type **25,40,71M** to move lines 25 through 40 in front of line 71. Type **3,3,7M** to move line 3 to line number 7. Type **,+7,3M** to move the current line and the next seven lines in front of line number 3.

If you want to practice moving lines, use the MEMO.TXT file you created previously. (If the version of MEMO.TXT you modified when you copied lines is still in memory, abort the session by typing **Q**, then restart EDLIN and load MEMO.TXT.) Type **L** at the EDLIN prompt to see the following listing of MEMO.TXT:

```
1:*MEMO TO: Scott N. Hollerith, President
2: From: Otto Wirk, Research Chief
3: Subject: New Dual Processor Micro
4: Date: Sunday
5: CC: Bill Kem, Marketing; Betty Site, Testing
6:
7: Our testing confirms that we should indeed
8: include two microprocessors in the next model
9: TLS-8E. This increases the odds that one will
10: work. If both happen to function, they can
11: take turns.
```

Type the following at the EDLIN prompt:

***3,3,5M**

This command moves line 3 to a position preceding line 5. Type **L** at the EDLIN prompt to confirm this. The memo now looks like this:

```
1:MEMO TO: Scott N. Hollerith, President
2: From: Otto Wirk, Research Chief
3: Date: Sunday
```

```
 4:* Subject: New Dual Processor Micro
 5: CC: Bill Kem, Marketing; Betty Site, Testing
 6:
 7: Our testing confirms that we should indeed
 8: include two microprocessors in the next model
 9: TLS-8E. This increases the odds that one will
10: work. If both happen to function, they can
11: take turns.
```

Q—The Quit Edit Command

Use the **Q** (Quit Edit) command to exit from EDLIN without saving the edited file. The syntax for this command is as follows:

Q

Use **Q** when you want to abort and terminate the edited file. EDLIN verifies whether you really want to abandon the edited file by displaying the following prompt:

```
Abort edit (Y/N)?
```

Press Y to abort the editing process or N to continue editing the file.

EDLIN does not create a backup when you are creating a new file. Backup files are created only when an existing file is being edited. When you use the **Q** command, a backup file is not created and the original file is not altered. If you are working on a new file and issue the **Q** command, the new file is lost.

Reminder:
*The **Q** (Quit Edit) command does not save a backup version of the current file.*

S—The Search Text Command

Use the **S** (Search Text) command to search and locate a sequence of text characters, called a *character string*. The syntax for the **S** command is as follows:

sline,*eline*?**Sstext**

sline is the starting line, and *eline* is the ending line of the range of lines to be searched. The *?* parameter produces the prompt `O.K.?` so that you can confirm whether you want to continue searching. If you answer Y to this prompt, the displayed line becomes the current line and the search ends. Answering N causes the search to continue. **stext** is the character string for which to search.

To search for text, type the starting line number of the text block to be searched. Press comma (,) and type the ending line number of the text block to be searched. Press the space bar and ? to have EDLIN prompt whether you want to stop searching. If you include this parameter, EDLIN queries you whether it should stop searching for the stext character string. To finish the

Reminder:
Use the ? parameter to have EDLIN prompt you to continue the search.

command, type **S** and the text string for which you want to search. Press Enter to start the search.

If the ending line is not specified, searching continues to the end of the file. If both starting and ending parameters are unspecified, searching begins at the line following the current line and continues to the last line of the file. The last line searched that contains the **stext** character string becomes the current line. Press Esc to cancel this command.

Reminder: *The last line in which the search text is found becomes the current line.*

Use the ? parameter to have EDLIN query you whether it should stop searching for the specified text string. Press Y to stop searching for text; the last line displayed becomes the current line. Pressing any other key continues the search. If you don't use the ? parameter, the text search stops on the first occurrence of the specified text string. You must issue another **S** (Search Text) command to continue the search.

If the **stext** string is not specified, EDLIN uses the value specified in the last **S** (Search Text) or **R** (Replace Text) command for the current session. When the specified text string cannot be found within the specified range of lines, the following message appears:

```
Not found
```

Reminder: *Remember that search strings are case-sensitive.*

Remember that search strings are case-sensitive. If you specify a search string of *capital*, for example, you will not find *Capital* or *CAPITAL*.

R—The Replace Text Command

Use the **R** (Replace Text) command to replace a sequence of text characters, a *character string*, with a different character string. The syntax for the **R** command is as follows:

sline,*eline*?**Rstext**<**F6**>**rtext**<**D**>

sline is the starting line of the search, and *eline* is the ending line of the search. The *?* parameter causes the prompt `O.K.?` to appear so that you can confirm each replacement by pressing Y or N. If you answer Y to this prompt, the replacement is made. Answering N causes the search-and-replace operation to continue, although no replacement is made at the current location. **stext** is the character string to search for, <**F6**> is the F6 key, and **rtext** is the replacement text.

To replace text, type the starting line number of the range of text to be affected by the replace operation. If you are replacing only one line, enter that line number. If you don't type a starting line number, EDLIN will start with the current default line. Press the comma key (,) and type the ending line number of the range.

At this point, you can press ? to have EDLIN prompt whether you want to change the displayed line. If you include this parameter, EDLIN queries you for each line that matches the search criteria. To continue with the command, type **R** and the text string for which you want to search. Then press the F6 key and type the text you want to use as the replacement. Start the command by pressing Enter.

Reminder:
Use the ? parameter to have EDLIN query whether you want to replace a particular instance of the search text.

To practice replacing text, load a fresh copy of MEMO.TXT. (If the version of MEMO.TXT you modified when you moved lines is still in memory, type **Q** to abort this EDLIN session. Restart EDLIN with the disk version of MEMO.TXT.) When MEMO.TXT is loaded, type this command at the EDLIN prompt:

***1,11?R TLS-8E<F6>TLS-8F/B1.8<D>**

This command tells EDLIN to search from lines 1 to 11 for the string `TLS-8E`. Because you have included the question mark, EDLIN asks you before replacing each occurrence of the string it finds. If you press Y when prompted, the new string, `TLS-8F/B1.8`, is inserted in place of the old one. List the memo to confirm that this has taken place:

```
1:MEMO TO: Scott N. Hollerith, President
2: From: Otto Wirk, Research Chief
3: Subject: New Dual Processor Micro
4: Date: Sunday
5: CC: Bill Kem, Marketing; Betty Site, Testing
6:
7: Our testing confirms that we should indeed
8: include two microprocessors in the next model
9:* TLS-8F/B1.8. This increases the odds that one will
10: work. If both happen to function, they can
11: take turns.
```

In this example, the current line becomes the line last replaced. If the starting line number is not specified, searching begins with the line following the current line. If the ending line is not specified, searching ends with the current default line. If both starting and ending line parameters are unspecified, searching begins at the line following the current line and continues to the last line.

Use the ? parameter to have EDLIN query you whether the displayed line should be replaced. Press Y to replace the text as specified in the command line. Pressing any other key prevents changes to the displayed line, and searching continues. Use the **S** (Search Text) command to search for text without replacing it.

If you do not specify the *rtext* parameter when you type the text parameters, all occurrences of the *stext* string are deleted (replaced with nothing) from the line specified. If you typed **1,4,Rmodel**, for example, any occurrence of the word *model* is deleted from the lines 1 to 4. If you don't specify either *stext* or

rtext, EDLIN uses the values specified in the previous **R** or **S** (Search Text) commands for this session.

In some cases, memory-resident programs may reassign the value of the F6 function key. This sometimes occurs when a keyboard or control program has reassigned keys. If you have a memory-resident program that has reassigned the function keys, use Ctrl-Z as an alternative to F6 if F6 is not available to you.

T—The Transfer Lines Command

The **T** (Transfer Lines) command transfers or merges the contents of a file into the file currently being edited.

Reminder:
When you use the ***T*** *(Transfer Lines) command, make sure that the file you are merging is an ASCII text file.*

Before using the **T** command within EDLIN, check to see whether the file being merged is an ASCII file. At the DOS prompt, use the DOS TYPE command to see whether the file is readable. Make sure that the file being merged is in the same directory as the file being edited. The syntax for the **T** (Transfer Lines) command is as follows:

*n***T***d:***filename.ext**

n is the line number in the current file where the new file is to be merged, *d:* is the drive designation for the file to be merged, and **filename.ext** is the name of the file to merge.

The **T** command merges files. To transfer lines, type the line number where the merged file contents are to be inserted. The current line is used if you don't specify a line number. Type **T** and type the drive name for the file to be merged (for example, A:, B:, and so on). You cannot designate a directory name; this is probably the most serious drawback of the **T** command.

Type the name of the file to be merged. Press Enter. The cursor returns to the * prompt, and the contents of the merged file are inserted in front of the current default line or the specified line number.

Summary

This chapter has introduced you to the commands and functions of the DOS-provided text editor, EDLIN. Following are the important concepts discussed:

- EDLIN is especially useful for editing and working with short files. Running EDLIN is often quicker and easier than loading a full-scale word processor.
- If you routinely use large files, you may want to invest in a full-screen editor or word processing program.

- You load EDLIN by typing **EDLIN** and the file name to be created or edited.
- You can use the function keys with EDLIN to reenter and correct text lines.
- You can view EDLIN files by using the **L** (List Lines) or **P** (Page) command.
- To edit files larger than 64K, use the **W** (Write Lines) command.
- The **I** (Insert Lines) command enables you to insert lines in a new or existing file.
- EDLIN's **R** (Replace Text) and **S** (Search Text) commands enable you to quickly look for and change specific strings of text.
- You can move and copy lines quickly by using EDLIN's **M** (Move) and **C** (Copy) commands.
- To abort editing changes, exit EDLIN by using the **Q** (Quit Edit) command
- To save your changes, use the **E** (End Edit) command.

In the next chapter, you learn the basics of creating batch files.

11

Understanding Batch File Basics

The idea of getting a computer to do work in convenient, manageable batches predates the personal computer by several years. The very first computers were large and expensive, and they could do only one thing at a time. But even these early machines were very fast. Making them wait to be told what to do next was inefficient.

Batch processing was developed to make computers more productive. Collections of tasks to be carried out consecutively were put together off-line, that is, without using the computer's resources. The chunks or batches of tasks were then fed to the computer at a rate that kept the computer busy and productive.

Today, computers are less inexpensive than precious human resources. Batch processing is still used to get the computer to carry out a series of tasks without wasting a lot of personnel time typing often-used or complex commands.

DOS has a host of special commands, called *subcommands*, that are used most often in batch files. Although you can type these commands at the DOS prompt, their primary application is to let you carry out other DOS functions in a more useful way. Batch subcommands are explained in this chapter and in Chapter 13.

Batch files are available for your convenience. After a few experiments, you probably will find this DOS facility quite handy. If you have DOS V4, you will learn in Chapter 13 how to install batch files as menu selections from the DOS Shell.

This chapter discusses the following batch subcommands:

- ECHO
- PAUSE

- REM
- GOTO
- CLS

Using Batch Files

A batch file is a simple ASCII file (a file consisting of ASCII text) containing a series of DOS commands. Many of the commands appear in the batch file exactly as if you had typed them at the DOS prompt. A simple batch file can consist of the following three lines:

```
DIR A:
DIR B:
DIR C:
```

When DOS executes this batch file, it first displays the directory of drive A:, then drive B:, and then drive C:. You can call the file DIRALL.BAT; whenever you type **DIRALL** at the DOS prompt, the directories of drives A:, B:, and C: are displayed. The three separate DOS commands are collected into a single batch file and executed with the "command" **DIRALL**.

Cue:
Use batch files to perform a single command or hundreds of commands.

Batch files have several advantages. As you have seen, once you place correctly phrased commands in a batch file, you can direct DOS to execute the commands by typing the batch file's root name. You use a batch file to direct DOS to perform one or hundreds of commands.

Another advantage is that once a batch file is invoked, DOS does not need your attention until the batch file is finished running. This capability comes in handy for programs that take a long time to run. For example, you can include the following commands in a batch file you use to copy files from one hard disk to another:

```
XCOPY C:\WP D:\WPBAK
XCOPY C:\DATA D:\DATABAK
XCOPY C:\UTILS D:\UTILBAK
```

Cue:
Create a batch file that does several tasks sequentially while you do something else.

The actual copying process may take a few minutes for each command. If you issue three separate XCOPY commands from the DOS prompt, you must wait until the first copy is finished before starting the second. If you put these three commands in a batch file, however, just typing the batch file name invokes all three commands when appropriate; you don't have to do anything else. While the batch file is running, you can have a snack, make a phone call, or do something else useful.

If the operation you want to perform requires your input to change disks, however, a batch file cannot perform this task. You must make allowances, by using

batch subcommands, for providing the correct disk if you want to use the batch file of XCOPY commands just described. You will learn more about these commands later in this chapter.

Understanding the Basic Rules of Batch Files

Batch files must follow certain special rules. A complete list of these rules is found at the end of the chapter. This section discusses some of the more prominent rules.

Cue:
You can use EDLIN, COPY CON, or your word processor in text mode to create an ASCII file.

Because a batch file contains only ASCII text, you can use the DOS COPY CON command, the DOS EDLIN line editor (described in Chapter 10), or another text editor to create a batch file. If you use a word processing program, make sure that the program is in programming (sometimes called nondocument or text) mode.

Reminder:
Batch files must have the extension .BAT.

Batch file names conform to normal DOS naming rules. The root name of a batch file can be from one to eight characters long and must conform to the rules for file names. The batch file must have the extension .BAT.

When DOS interprets a command you type, it disassembles, or *parses*, the line to look for commands it recognizes. DOS finds internal commands, like DIR and COPY, if the commands are typed with the correct syntax (that is, **DIR A:** rather than **A: DIR**). If the line includes a command that is not internal, DOS searches for a system file with a root name matching the command. If you type **FORMAT**, DOS looks for FORMAT.COM or FORMAT.EXE (the extension varies with your DOS version) in the current directory as well as the directory defined by the PATH statement. DOS looks for these files in this order: .COM files first and then .EXE files. If DOS does not find a .COM or .EXE file matching the command, it looks for the third type of system file, the one with the .BAT extension.

Reminder:
DOS searches for command files in this order: internal, external .COM, external .EXE, and external .BAT.

Because of the order of this search, you cannot easily give a batch file the same root name as a DOS internal or external command. Nor can you give the same name as another .COM or .EXE file found in the current path. (You can, however, invoke a batch file named FORMAT.BAT by typing the full path name explicitly: **C:\BATCH\FORMAT.BAT**, for example).

Creating Simple Batch Files

For practice, create a few simple batch files to see what they can do. In each of the following examples, use EDLIN, COPY CON, or your word processor to create the files in plain ASCII text.

Creating a Batch File To Format Diskettes

The first example demonstrates how batch files can keep you from having to learn seldom-used syntax. Assume that your computer is equipped with a 3 1/2-inch, 1.44M high-density drive. Occasionally, you want to format a 720K double-density disk so that you can exchange files with a co-worker who has a computer that cannot read high-density disks.

Caution: Don't format double-density disks with high-density format, or you may lose data.

Some computer systems can automatically format a 3 1/2-inch double-density disk for 720K. These computers have a sensor that detects the extra "hole" in 1.44M high-density disks, and does not use the denser format unless the hole is found. IBM PS/2 computers don't have this sensor; you can format any 3 1/2-inch disk for 1.44M. You may regret formatting double-density disks with high-density format, however. Double-density disks almost always will take a high-density format, and you can store 1.44M of data on disks formatted in this way. Weeks or months later, however, when you want to retrieve that information, you may find that reading the disk is a problem. An alarmingly high percentage of double-density disks lose data when formatted for higher density. Don't do it, unless saving a few dollars is more important than your data.

The reverse technology, however, is perfectly safe. DOS has a way to format high-density 1.44M disks for double-density 720K use. As you learned in Chapter 5, versions of DOS after V3.3 enable you to specify track density (as well as other factors). With DOS V3.3, you can enter the number of sectors per track and the number of tracks. DOS ignores inappropriate parameters, however; you can't specify 13 sectors or 43 tracks, for example. DOS V4 simplifies this by enabling you to enter a size parameter, such as 720, 720K, 1.44, and the like.

Because you may not remember the syntax when you want to format a disk, you can create a batch file to do the job for you. Create an ASCII file called 720.BAT. Type each of the following lines on a separate line in the file:

```
ECHO OFF
ECHO Insert diskette to be formatted for 720K in Drive A:
ECHO Press Enter when ready.
PAUSE>NUL
FORMAT A: /N:9 /T:80
```

This first batch file is relatively simple so that you can understand the function of each command more easily. Each of the lines in the batch file is described in the following paragraphs.

ECHO OFF

Cue: Use ECHO OFF to turn off the display of commands in the batch file.

The first line of the batch file is ECHO OFF. ECHO is a DOS command related to the BASIC PRINT or DOS TYPE command. Instead of displaying a file on-screen (as does TYPE), ECHO displays the characters that follow ECHO. ECHO can be used to turn ON or OFF the display of DOS commands being carried out by the batch file.

The line ECHO OFF turns off the on-screen display of the batch file's commands. Once ECHO OFF is invoked, you don't see any of the other commands on-screen; you only see the results of the commands. Because ECHO is ON when you start 720.BAT, you see the first command, ECHO OFF, displayed on-screen. If you have DOS V3.3 or later, you can eliminate even that display by placing the @ symbol before the command:

@ECHO OFF

Cue:
With DOS V3.3 or later versions, you can use the @ symbol to turn off the display of the command that follows it.

The @ tells DOS not to display the command that follows. Turning ECHO OFF at the beginning of a batch file makes the screen easier to read; only the information you want to be displayed is actually shown.

The next two lines of the batch file are an example of another use for ECHO—to display text on-screen.

ECHO Insert disk to be formatted for 720K in Drive A:
ECHO Press Enter when ready.

Reminder:
The ECHO command prints on-screen the characters that follow it.

In this case, the text is a message to the user, explaining what is about to take place and requesting that Enter be pressed when the disk is ready. You can use ECHO for prompts or to display other information in your batch files.

PAUSE>NUL

PAUSE is another batch-file subcommand. PAUSE causes the batch file to suspend execution and displays this message on-screen:

```
Strike a key when ready ...
```

This all-purpose message is less than useful for a number of reasons. First, you may want the user to do something other than press any key when ready. For example, batch files can be stopped entirely by pressing Ctrl-C or Ctrl-Break, so you may want to provide the user with a choice of keys to press, such as "Press a key to proceed with reformat of drive C:, or press Ctrl-C to abort." In addition, you may not like the word *strike*. Some users may take the "strike" suggestion too literally. You may prefer to use the word *press*.

Although the term "a key" implies you can press *any* key in response to the PAUSE command, some keys do not work. The Caps Lock, Shift, and Ctrl keys are not acceptable entries to the PAUSE command. For accuracy, the PAUSE message should say, `Strike an alphanumeric, function, cursor, Esc, Ins...` The ECHO line above the PAUSE command asks the user to press Enter, not because that particular key is required, but because the Enter key is large, easy to find, and avoids confusion that may result from the "press a key" terminology.

Cue:
*Use **PAUSE>NUL** if you want the computer to pause without displaying the default message.*

The redirection symbol > following the PAUSE command redirects the output of PAUSE (the `Strike a key when ready` message) to the NUL device. When you include PAUSE>NUL in the batch file, the PAUSE message does not appear

on-screen; the user sees only the message displayed by the preceding ECHO command: `Press Enter when ready.`

FORMAT A: /N:9 /T:80

This final line of the batch file does all the work. If you include this line in the batch file, you do not have to type the drive specification **A:** or the parameters that force the 720K format when you want to format a disk. All you have to do is type **720** at the DOS prompt and press Enter.

Creating a Batch File To Mark a Directory as "Home"

The next batch file also uses only the familiar DOS ECHO and PAUSE commands. This batch file, like the last file, uses redirection, but in a more useful way.

To use this batch file, you should have on your hard disk a directory called C:\DOS, in which all your DOS files (such as XCOPY.EXE) reside. You can substitute for C:\DOS the name of the directory containing your DOS files. The PATH statement in your CONFIG.SYS file should point to the directory that contains all the DOS files so that DOS can locate any files it needs, no matter which directory is current. (Read more about the PATH statement in Chapter 6.)

Create an ASCII file called MARK.BAT and type the following lines in the file:

```
ECHO OFF
CD>C:\DOS\HOME.ASC
COPY C:\DOS\CD.ASC+HOME.ASC C:\DOS\HOME.BAT
ERASE C:\DOS\HOME.ASC
```

In this batch file, the output of the DOS CD command is redirected to a file called HOME.ASC, which is located in the C:\DOS directory. As you may remember, the CD command, typed at the DOS prompt by itself, displays the name of the current directory. By redirecting that output to a file, you create a file called HOME.ASC that contains the name of the current directory.

The third line of the batch file combines the HOME.ASC file with another file called CD.ASC. The combined files are placed in a file called HOME.BAT. CD.ASC contains only the characters `CD` followed by a space. When you combine CD.ASC and HOME.ASC, the following line is placed in HOME.BAT:

`CD` *directoryname*

This command, of course, has the effect of changing from whatever directory you are in to the "home" directory marked by the MARK.BAT file. You can use the command **MARK** to invoke the MARK.BAT file to specify the current direc-

tory as the home directory. Then you can change directories (perhaps using batch files) to perform other tasks. To return to the home directory, type **HOME** to invoke the HOME.BAT file created by MARK.BAT. HOME.BAT returns you to the home directory (assuming that you didn't log over to another hard disk or to a floppy disk in the meantime).

Because HOME.BAT remains on disk, it "remembers" the home directory even if you turn off the computer. You can use MARK.BAT to mark a home directory for a short session or for longer use.

Before you use MARK.BAT, you must create the file CD.ASC. Follow these instructions carefully to create this file:

Step 1. *At the DOS prompt, type the following and press Enter:*

COPY CON C:\DOS\CD.ASC

If your DOS files are located in a directory other than C:\DOS, substitute the name of that directory.

Step 2. *Enter the following line:*

CD<space><F6><Enter>

Note: After you type **CD**, press the space bar, the F6 key, and Enter. Do not type the characters <space><F6><Enter>; also, do not press Enter before you press F6.

This process creates a file containing only the characters CD and a space, with no carriage return between them. When MARK.BAT combines CD.ASC and HOME.ASC, the text in each file is combined on a single line in the HOME.BAT file.

Understanding Batch File Parameters

You can create quite sophisticated batch files using only DOS commands and the simplest batch subcommands. The batch files created so far in this chapter carry out exactly the same functions every time you use them. You may want a batch file to perform a different operation each time you use it, however, even though the commands in the file are fixed. You can use the same batch file to do different things by taking advantage of batch file parameters.

Reminder: *Parameters are additional information included in the command and separated by delimiters.*

A review of parameters in general helps you understand batch file parameters. In DOS, many commands accept additional information when you issue the command. COPY is an example of such a command. The name of the file you want to copy, the destination to which you want to copy the file, and any additional switches you enter are all parameters. Parameters, then, are the additional information you type after you type the command name at the DOS

prompt. Commands (or command programs) can use whatever information is in the parameters.

Consider the following command:

COPY FORMAT.COM B: /V

Every word or set of characters separated by a space, comma, semicolon, or other valid delimiter is a parameter. FORMAT.COM, B:, and /V are parameters. Each parameter controls some aspect of the COPY command. In this example, the parameters tell COPY to copy the file FORMAT.COM from the default disk to the disk in drive B: and to verify that DOS has correctly made the copy.

Cue:
Use variable markers to make your batch files work with parameters.

Batch files also use parameters, but in a different manner from commands. When you type the name of the batch file at the DOS prompt, you can add up to nine parameters. DOS assigns a variable name to each of these parameters. You can use the variable name in your batch file in the form of a variable marker numbered from 0 to 9. You precede the marker number with a percent sign to inform DOS that you are specifying a replaceable parameter.

Understanding More About Variable Markers

This section examines how variable markers can be used in a batch file. Assume that you have a batch file called DELETE.BAT. (The DOS commands to erase a file are ERA, ERASE, and DEL. Because no DELETE command exists, you can use the word as a batch file name.) Suppose that the batch file DELETE.BAT looks like this:

```
ECHO ON
ERASE %1
ERASE %2
ERASE %3
ERASE %4
ERASE %5
ERASE %6
ERASE %7
ERASE %8
ERASE %9
```

When you invoke DELETE, suppose that you follow the batch-file name with the names of some files to be removed, as shown by the following example:

DELETE TEST.BAK OLDFILES.* C:\BAK\JUNK.DOC

This example specifies three parameters, all of which are file specifications. TEST.BAK refers to a file in the current directory; OLDFILES.* specifies all files in the current directory with the root name OLDFILES and any extension; C:\BAK\JUNK.DOC refers to the specific file JUNK.DOC in the C:\BAK subdirectory.

When the second line of the batch file is called, DOS substitutes for %1 the first parameter you typed at the DOS prompt: TEST.BAK. Because ECHO ON enables you to see the commands, the screen shows the following:

```
ERASE TEST.BAK
ERASE OLDFILES./*
ERASE C:\BAK\JUNK.DOC
```

You may have several questions at this point: What happens when the fifth line is executed? It calls for the fourth parameter %4, but none was typed. Also, why isn't %0 used as a variable marker?

When a batch file calls for a parameter that isn't typed as part of the command, DOS substitutes nothing for the variable. In this case, DOS tries to carry out six more ERASE commands that, because no parameters for the variable markers %4 through %9 were specified, display on-screen as shown:

Caution:
When a batch file calls for a parameter that isn't typed as part of the command, DOS substitutes nothing for the variable.

```
ERASE
ERASE
ERASE
ERASE
ERASE
ERASE
```

DOS reports `Invalid number of parameters` and aborts the batch file at that point. The lesson here is that you must account for all the parameters used in your batch file. Fortunately, DOS provides easy ways of doing that.

Do not start your list of variable markers with %0 unless you want DOS to substitute the first thing you type at the DOS prompt—the batch file name itself—for %0. In this example, the %0 variable contains DELETE.BAT. Although the %0 variable was not used in this batch file, you may want to use it in some instances. If you created a batch file that called itself, for example, you would want to be able to call the file even if it happened to rename itself. Suppose that you included the following line in the DELETE.BAT file:

Cue:
Use the %0 variable marker if you want to use the batch file name itself as a parameter.

ECHO This batch file is called %0

The batch file could be renamed as DELETE.BAT, DELETER.BAT, or ERASER.BAT; the proper name would be substituted whenever %0 was used.

If you want further practice including parameters in a batch file, use a text editor or the COPY CON technique to create a file called TEST.BAT. Type the following line into the file:

ECHO Hello, %1

Now type **TEST**, press the space bar, type your first name, and press Enter. The results should display on-screen like this:

```
C>TEST DAVID
C>ECHO Hello, DAVID
Hello, DAVID
C>
```

To see how DOS replaces the parameters, create another batch file called TEST1.BAT. Type the following line into this file:

```
ECHO %0 %1 %2 %3 %4
```

After you create this file, type **TEST1**, a space, your name, another space, and your street address. Following is the screen display:

```
C>TEST1 DAVID 1234 PIONEER STREET
C>ECHO TEST1 DAVID 1234 PIONEER STREET
TEST1 DAVID 1234 PIONEER STREET
C>
```

The batch file command instructs DOS to display on-screen the parameters 0 through 4. These parameters worked out to be the following:

Word:TEST1DAVID1234PIONEERSTREET

Parameter:%0%1%2%3%4

Now try to "shortchange" DOS again by not specifying a sufficient number of parameters to fill each expected variable marker. Run TEST1 again, but this time give only your first name. You should see something like the following:

```
C>TEST1 DAVID
C>ECHO TEST1 DAVID
TEST1 DAVID
C>
```

DOS displayed the batch file name and your first name. No other information was echoed to the screen. Although not enough parameters were specified, DOS replaced the unfilled markers with nothing. In this case, unlike the ERASE example, the empty markers did no harm.

Constructing a Batch File Using Parameters

In this section, you will learn how to construct a batch file that takes advantage of parameters. Suppose that you use several computers, but one of the hard disk systems is your "workhorse" where you store the files you want to keep.

Suppose that you use diskettes to move information between computers. You usually delete the file from the diskette after the file has been copied back to a hard disk. You use the following steps to transfer data from a floppy disk to a hard disk:

Step 1. *Copy the file from the diskette to the hard disk.*

Step 2. *Erase the file from the diskette.*

To simplify this process, you can create a batch file called C&E.BAT (copy and erase). Type the following commands in the file:

```
COPY A:%1 C:%2 /V
ERASE A:%1
```

To use the C&E.BAT file, type the following at the DOS prompt:

C&E*oldfilename newfilename*

The first parameter, *oldfilename*, represents the name of the file you want to copy from the floppy disk to the hard disk; *newfilename* is the new name for the copied file (if you want to change the file name as it is being copied).

Suppose that you put a diskette containing the file NOTES.TXT into drive A: and want to copy the file to the hard disk. Type the following at the DOS prompt:

```
C&E NOTES.TXT
```

Here's what the screen looks like:

```
C>COPY A:NOTES.TXT C: /V

1 file(s) copied

C>ERASE A:NOTES.TXT
C>
```

Notice that, even though you didn't type the parameter *newfilename*, DOS carried out the batch file, keeping the same file name during the copy. DOS copied NOTES.TXT from drive A: to drive C: and then deleted the file on the diskette in drive A:. The %2 parameter was "dropped," and the file did not get a new name when it was copied.

Cue:
Use a parameter intended for the file's new name as a directory name.

One of the benefits of constructing a batch file this way is that you can use a path name as the second parameter. By specifying a path name, you can copy the file from the floppy disk to a different directory on the hard disk. To copy NOTES.TXT to the directory called WORDS, for example, type the following:

```
C&E NOTES.TXT \WORDS
```

The screen display is as follows:

```
C>COPY A:NOTES.TXT C:\WORDS /V

1 file(s) copied

C>ERASE A:NOTES.TXT
C>
```

Because \WORDS is a directory name, DOS knows to copy the file NOTES.TXT into the directory \WORDS. DOS does not give the new file the name WORDS.

Using Additional Batch Subcommands

This section provides additional information about several more subcommands that are useful when you are working with batch files.

Introducing REM and More about ECHO

Reminder:
If you specify a REM statement of more than 123 characters, DOS truncates the statement.

As you learned earlier in this chapter, the ECHO command does two things: ECHO ON and ECHO OFF turn on and off the display of lines from batch files as the commands are executed. ECHO is also used to display messages.

When ECHO is used to display messages, the command parallels the REM (remark) command, which also displays messages on-screen. ECHO can display a message with a maximum length of 122 characters; REM can display 123 characters (the 127-character DOS command-line limit, minus the length of the command, minus a space).

If you include in a batch file a REM statement longer than 123 characters, DOS simply doesn't display the excess characters. Keep this in mind if you need a long remark and don't care whether the entire remark is displayed. Indeed, REM is rarely used to display messages on-screen.

The ECHO setting affects REM. If ECHO OFF is set, no REM message appears on-screen. If ECHO ON is set, REM messages are displayed. A message following the ECHO command, on the other hand, always appears on-screen, regardless of whether ECHO OFF has been issued. If you always want a message in your batch file to appear, use ECHO rather than REM.

Cue:
Use REM for comments you don't want to appear on-screen; use ECHO for messages you want to display.

You can put both ECHO and REM to good use in a batch file. When you review a batch file some time after you create it, you may have forgotten why you used certain commands, or why you constructed the batch file in a particular way. Leave reminders in your batch file with REM statements. Comments in REM statements don't appear on-screen when you issue the ECHO OFF command, and the comments make the batch file self-documenting, which you and other users will appreciate later. If you want the batch file to display particular messages, use ECHO. Messages set with ECHO appear on-screen whether or not you set ECHO OFF.

Cue:
Use a colon at the beginning of a line to indicate a remark or a label.

Another way to include a remark in a batch file is by using a colon. If the first character on a line in a batch file is a colon, DOS assumes that the line is a label and does not try to carry out any commands on that line. Using colons, however, may confuse DOS into thinking that "remark" is a true label (labels are

described in a later section). You must be certain that the first eight characters of a statement you want interpreted as a remark don't match an actual label used elsewhere in the file. The distinction between remarks and labels becomes more clear later in this chapter.

You can use the colon to provide a second type of remark—one not displayed under any circumstances, whether or not ECHO OFF is set. You can use the colon to mark sections of the batch file where ECHO ON has been set but where you still do not want to display any REM statements. You also can use the colon to temporarily "remove" portions of the file you don't want DOS to carry out. You may want to mark sections of the file in this way when you debug or test the file.

Reminder: *Use the colon to indicate a line never to be displayed on-screen or to mark a line not to be executed.*

This use for the colon as a remark is not a "standard" DOS practice, but is widely used nonetheless. If you keep in mind the potential danger of confusing DOS with labels and remarks, you can use the colon safely to enhance your batch files.

Controlling the Display with CLS

The DOS CLS command can be used at any time at the DOS prompt to clear the screen. In batch files, CLS is used commonly after the ECHO OFF command. The reason CLS follows the ECHO OFF command is that the ECHO OFF command itself appears on-screen. To keep the screen clear, issue the CLS command as shown:

Cue: *Use CLS to clear the screen completely.*

```
ECHO OFF
CLS
```

With DOS V3.3 or later versions, you can use the @ feature to prevent the ECHO OFF command from appearing on-screen. Replace the preceding two command lines with the following single line:

```
@ECHO OFF
```

Because @ is the first character on the line, ECHO OFF does not appear on-screen, and the ECHO OFF command stays in effect until the end of the batch file. Using the @ feature does not clear the screen, however. If you want to clear the screen when you start a batch file, use the CLS command in the batch file.

Branching with GOTO

The DOS GOTO batch subcommand is similar to the GOTO command in the BASIC programming language. With the DOS GOTO command, you can jump to another part of your batch file. The DOS GOTO command uses a label, not a

line number, to specify the destination. A *label* is a batch-file line that starts with a colon and is followed by a one-character to eight-character name. The label name can be longer than eight characters (as when you use the colon to put comments in your file or to "comment out" commands you want to bypass), but only the first eight characters are significant as a label.

When DOS encounters a GOTO command in the batch file, DOS starts at the beginning of the batch file and searches for a label matching the one specified by GOTO. DOS then jumps to the line in the batch file following the line with the label.

As an example, the batch file TEST2.BAT is listed here. This file is similar to the TEST.BAT batch file you created previously, with the addition of the GOTO and PAUSE subcommands.

```
:START
ECHO Hello, %1
PAUSE
GOTO START
```

If you invoke the batch file by typing **TEST2 DAVID**, the screen shows the following:

```
C>TEST2 DAVID
C>ECHO Hello, DAVID
Hello, DAVID
C>PAUSE
Strike a key when ready... <space>
C>GOTO START
C>ECHO Hello, DAVID
Hello, DAVID
C>PAUSE
Strike a key when ready...^C
Terminate batch job (Y/N)? Y
C>
```

Cue:
GOTO can cause an infinite loop that can be broken by pressing Ctrl-C or Ctrl-Break.

The batch file began by echoing the message that included the name DAVID. The file then paused so that you could press a key before it continued. After you pressed a key (in this example, the space bar), DOS executed the `GOTO START` command. DOS then jumped to the line following `:START` and continued. When DOS paused a second time in this example, Ctrl-C was pressed to stop the batch file. DOS asked whether you wanted to stop the batch file. In this example, the response was Y for yes, so DOS stopped processing the batch file and returned to the system prompt. This type of batch file causes DOS to loop continuously until you type the Ctrl-C or Ctrl-Break sequence.

You can use an endless loop to repeat the batch file several times. You also can repeat the batch file a different number of times each time you invoke the file.

For example, suppose that you want to copy the files CHKDSK.COM, FORMAT.COM, DISKCOMP.COM, and DISKCOPY.COM to several disks. You may want to copy these files onto one, two, or a hundred disks. With a batch file, you can make DOS do the hard work.

For this example, assume that the computer has two disk drives. The diskette that contains the files CHKDSK.COM, FORMAT.COM, DISKCOMP.COM, DISKCOPY.COM, and the new batch file is in drive A:. The diskette to receive the files is in drive B:.

First, create a batch file called FLOPPY.BAT. Enter the following statements in the file (remember that the @ symbol in line 1 can be used only with DOS V3.3 and DOS V4). Do not type the line numbers at the beginning of each line; these numbers are references only.

```
(1) @ECHO OFF
(2) :START
(3) ECHO Place the disk to receive CHKDSK.COM, FORMAT.COM,
(4) ECHO DISKCOMP.COM, and DISKCOPY.COM in drive B:.
(5) ECHO To quit, type Ctrl-C and press Y, or
(6) PAUSE
(7) COPY A:CHKDSK.COM B: /V
(8) COPY A:FORMAT.COM B: /V
(9) COPY A:DISKCOMP.COM B: /V
(10) COPY A:DISKCOPY.COM B: /V
(11) ECHO Files are copied.
(12) GOTO START
```

In this file, the GOTO command on line 12 causes an endless loop. The file also exhibits several batch file subcommands.

Line 1 sets ECHO OFF so that the commands are not displayed. The @ sign at the beginning of the line stops the ECHO OFF line itself from being displayed. Lines 2 through 5 tell the user of the batch file either to put a diskette into drive B: or to press Ctrl-C to quit.

The PAUSE command on line 6 displays the `Strike a key when ready...` message. If you have finished copying files when this prompt appears, press Ctrl-C and answer Y to the `Terminate batch job (Y/N)?` prompt.

Lines 7 through 10 actually copy the four programs to the disk in drive B:. Line 11 is the "reassurance" line. It tells you that the files have been copied. This line is not necessary, but it assures you that all has gone well. Line 12 is the GOTO line that starts the process over again.

Notice the arrangement of lines 2 through 5. The on-screen result is as follows:

```
Place the disk to receive CHKDSK.COM, FORMAT.COM,
DISKCOMP.COM, and DISKCOPY.COM in drive B:.
To quit, type Ctrl-C and press Y, or
Strike a key when ready...
```

Cue: You can incorporate the PAUSE message into a string of ECHO commands.

The `Strike a key when ready` message is generated by the PAUSE command. Using ECHO to create your own message that duplicates the PAUSE message isn't necessary. Instead of inserting a customized message, this file depends on the PAUSE message to tell you to strike any key when ready. You can, however, include the line PAUSE>NUL o redirect the PAUSE message to the NUL device and then supply your own message.

Reviewing Batch File Rules

This chapter has explored the basics of understanding batch files. Additional batch file commands are explained in Chapter 13. You have learned enough about batch files to use a brief list of rules that apply to batch files. A few of these rules require some additional explanations found in Chapter 13 before you can really put the rules to work.

Rules for Batch Files

1. If you do not specify a disk-drive name when starting a batch file, the current disk drive is used.
2. If you do not specify a path, the current directory is used.
3. To invoke a batch file, type the root name and press Enter. If you have a file with the same root name and the extension .COM or .EXE in either the current directory or the path, you must include the .BAT extension when invoking the batch file, or DOS runs the .COM or .EXE file.
4. If the batch file is not in the current directory of the disk drive, and you did not specify a path name, DOS searches the directory or directories specified by the PATH statement to find the batch file.
5. DOS executes each command in the batch file one line at a time. Any specified parameters are substituted for the variable markers in the file.
6. DOS recognizes a maximum of ten parameters. You may use the SHIFT subcommand to get around this limitation. SHIFT is explained in Chapter 13.
7. If DOS encounters an incorrectly phrased command when running a batch file, the message `Syntax error` is displayed. DOS ignores the rest of the commands in the batch file, and the system prompt reappears.

8. You can stop a batch file by pressing Ctrl-Break or Ctrl-C. DOS displays the following message:

   ```
   Terminate batch job (Y/N)?
   ```

   If you answer Y, the rest of the commands in the file are ignored, and the system prompt appears. If you answer N, DOS skips the current command but continues processing the other commands in the file.

9. DOS remembers which directory holds the batch file. Your batch file may change directories at any time.

10. Most versions of DOS remember which disk drive holds the batch file, so you can change disk drives freely. DOS V3.0 does not remember which disk holds the batch file, however. If you remove the disk that holds the batch file, DOS V3.0 displays an error message. DOS V3 and V4 request that you reinsert the disk holding the batch file and press a key to continue.

11. You can instruct one batch file to run another batch file after the first batch file is finished. When you enter the name of a second batch file as the last command in the first batch file, control of the computer passes to the second batch file. If the COMMAND or CALL command is used to call the second batch file, the second batch file is executed, and control returns to the first batch file. Both COMMAND and CALL are discussed in Chapter 13.

Summary

Batch files can make your computer do the hard work for you. They can replace repetitive typing with commands executed automatically. As you work with batch files, remember the following key points:

- ❑ You must give batch files the extension .BAT.
- ❑ You invoke batch files by typing the root name of the batch file and pressing Enter. You may specify an optional disk drive name and path name in front of the batch file name.
- ❑ You can use in a batch file any command you can type at the DOS prompt.
- ❑ You can use the ECHO subcommand to turn on or off the display of DOS commands being executed by the batch file.
- ❑ The PAUSE subcommand causes the batch file to suspend execution and displays a message on screen.

- ❑ Use the REM subcommand to leave comments and reminders in your batch file; the comments do not display on-screen.
- ❑ Each word (or set of characters) in a command separated by a delimiter is a parameter. When you use a batch file, DOS substitutes the appropriate parameters for the variable markers (%0 to %9) in the file.
- ❑ The GOTO subcommand can be used to create a loop in the batch file.
- ❑ Use the CLS subcommand to clear the screen completely.

In the next chapter, you learn more about ANSI.SYS, the device driver used to enhance your keyboard and display.

12

Understanding ANSI.SYS

The two parts of the discussion of batch files are separated by this chapter on ANSI.SYS for a simple reason: you can use the features of ANSI.SYS to perform many useful tasks with batch files. This chapter explains how to use the ANSI.SYS device driver with DOS; the next chapter explains some techniques for incorporating these capabilities into batch files.

ANSI.SYS is a device-driver file you can use to give DOS more elaborate control over the computer's display screen. ANSI.SYS "fits" between the keyboard and the CPU and between the CPU and the monitor to intercept special characters and act on those characters. If you don't use ANSI.SYS, DOS takes care of the display in a completely satisfactory manner. Using ANSI.SYS, however, enables you to control more directly what appears on-screen.

First, you need to learn about device drivers in general, and ANSI.SYS in particular.

Installing Device Drivers

As you have learned, DOS can work with many different peripherals or devices, including printers, hard disks, scanners, and mice. Even your keyboard and CRT display are treated as separate devices by DOS.

DOS and the ROM-BIOS of your computer can work with this array of devices adequately. As long as your printer accepts a standard set of commands, for example, you can connect it to the LPT or COM ports and issue the appropriate DOS commands. DOS also "knows" how to handle other common devices, such as the keyboard and display.

The operating system would be very limited indeed if it could "talk" with only the devices known at the time that version of DOS was issued. Moreover, requiring the programmers of the operating system to rework DOS every time a vendor came out with a new nonstandard device would be inefficient. Asking the vendors themselves to provide "patches" for DOS would be equally vexing.

Reminder:
Use device drivers to expand DOS's capabilities.

Instead, DOS enables you to specify the utilities called *device drivers* in a start-up file named CONFIG.SYS. CONFIG.SYS uses many different directives, such as BUFFERS and FILES. These directives are discussed in Chapter 14. This chapter concentrates on the DEVICE directive, or command, that enables you to install drivers like ANSI.SYS.

A device driver is a special, low-level program that interfaces between DOS and the device. A device driver accepts input and output for the device and "translates" the information so that DOS or the device can understand it. Sometimes a device driver may recognize special pieces of information and perform a particular action based on the special data.

ANSI.SYS is a special device driver supplied with DOS that works with the display and keyboard. ANSI.SYS provides additional capabilities beyond those built into DOS. For example, you can use ANSI.SYS to redefine keyboard definitions or to change the colors displayed on your screen. Although you don't actually change the ANSI.SYS file, you use directives to enable particular parts of the file to accomplish what you want to do. Once the file is activated, it is always ready to respond to your special commands.

Reminder:
If you have DOS V2.0 or later, you can use device drivers.

V2.0 was the first version of DOS to include provisions for device drivers. V2.0 opened the door to a whole world of new peripherals by simplifying the process of writing device drivers. Users, in turn, easily can install these prewritten drivers by including this simple directive in CONFIG.SYS:

DEVICE=*driver_filename*

Like a batch file, CONFIG.SYS is an ASCII text file containing a few commands or directives. CONFIG.SYS can be created with COPY CON, EDLIN, or your word processor (in text mode). DOS checks CONFIG.SYS while booting up, before it carries out the commands in AUTOEXEC.BAT. CONFIG.SYS has fewer legitimate directives than AUTOEXEC.BAT.

Reminder:
CONFIG.SYS must be in the root directory.

CONFIG.SYS must be located in the root directory of the disk you use to boot DOS. You may already have a CONFIG.SYS file. If so, load it into your favorite editor and examine it to see whether a line containing ANSI.SYS is included. If the line does not exist, you can add the following line to the file if you want to activate ANSI.SYS:

DEVICE=ANSI.SYS<F6>

Note: The <F6> in this directive is the same as the Ctrl-Z, or end-of-file, sequence.

If you don't have a CONFIG.SYS file, create one by typing the following at the DOS prompt:

COPY CON CONFIG.SYS

DEVICE=ANSI.SYS<F6>

In this example, ANSI.SYS is located in the root directory with CONFIG.SYS. You can locate ANSI.SYS in a subdirectory if you specify that directory in the DEVICE statement, as shown in the following example:

```
DEVICE=d:\directory_name\ANSI.SYS
```

In either case, CONFIG.SYS must be located in the root directory.

Communicating with ANSI.SYS

Once ANSI.SYS is loaded by CONFIG.SYS, all input from the keyboard and output to the screen are routed through ANSI.SYS. If you don't load ANSI.SYS, DOS uses simple, adequate input/output routines built into the DOS kernel to control the keyboard and the display. You can communicate with ANSI.SYS by modifying the streams of characters it receives and sends with special strings of characters. All ANSI.SYS commands begin with the escape character. Escape alerts ANSI.SYS to look for one of the special ANSI.SYS directives in the string that immediately follows.

Reminder: All ANSI.SYS commands begin with the escape character.

You cannot simply type **escape** from the keyboard, however. When you press the Esc key, DOS's default response is to cancel its current operation and move the cursor down to the next line. You have to send the escape character to ANSI.SYS in an encoded fashion using one of two simple methods.

Embedding Escape Characters in Files

The first way to enter the escape character is to embed it in a file, which you then can send to the screen with the TYPE command. When ANSI.SYS receives the escape character and the directive following it, ANSI.SYS carries out the directive as specified. Many word processors (and COPY CON) don't let you enter an escape character directly. EDLIN doesn't either, but does allow you to type an escape character as described in Chapter 10. To recap, you press Ctrl-V and the left square bracket ([). When these characters are entered, EDLIN shows something like this:

Reminder: If you use EDLIN to edit ANSI.SYS, enter an escape character by pressing Ctrl-V and [.

```
1:*^V[
```

When you later list the line, however, the *V* is not shown. The line looks like this with most versions of DOS:

```
1:*^[
```

With DOS V3.0 and V3.1, the bracket and caret are reversed:

```
1:*[^
```

Once the escape character has been entered into a file, save the file and use the DOS TYPE command to display the file on-screen. In the process of sending the characters in the file to the display, ANSI.SYS recognizes the escape character as a special character, holds that character (does not display it), and waits for another character or string of characters to complete the directive begun with escape. You can include the remaining characters in the file or complete the directive by typing the appropriate characters from the keyboard.

Using PROMPT To Enter Escape Characters

Reminder: *You can use the* ***PROMPT $e*** *command to enter an escape character.*

A second way to send an escape character to ANSI.SYS is with the DOS PROMPT command. You can use this command in a batch file or at the DOS prompt. The syntax for this command is as follows:

SET PROMPT string

or

PROMPT string

Normally, **string** is a set of characters you want to use as the DOS prompt, such as *By Your Command, Master!* However, DOS provides a group of special subcommands that PROMPT can use as arguments interpreted and acted on by ANSI.SYS. Many of these subcommands enable you to personalize your system prompt. These subcommands are variables replaced by DOS with the current disk drive letter, the subdirectory, and so forth. The subcommands consist of the dollar sign ($) and another character (see table 12.1).

Table 12.1
PROMPT Subcommands

Symbol	*Meaning*
$e	Escape character
$t	Time
$d	Date
$v	DOS version number
$n	Default drive name
$g	Greater than symbol
$l	Less than symbol
$p	Current drive and path
$b	Vertical bar
$q	Equal sign
$h	Backspace
$_	Underscore (line feed)

Reminder: *Use lowercase letters for PROMPT subcommands.*

You must use lowercase characters for the PROMPT subcommands. If you use PROMPT to send directives to ANSI.SYS, you must make sure the batch file turns ECHO ON, or ANSI.SYS never "sees" the characters you're sending. To send an escape character to ANSI.SYS using the PROMPT command, include a line like this in a batch file:

PROMPT $e

When you run the batch file, and DOS tries to display the characters specified by the PROMPT line, ANSI.SYS interprets the $e as the beginning of a special directive. You can append other subcommands to this same PROMPT line.

To summarize, you can alert ANSI.SYS that a command is being sent by preceding the command with an escape character. The escape character and command are sent to ANSI.SYS either by creating a file into which the escape character (Ctrl-V [) and command are entered and then using the DOS TYPE command with that file, or by using the PROMPT command to specify the escape character ($e). The following sections present the actual ANSI.SYS commands. The commands are divided into screen commands, cursor-movement commands, and keyboard commands. More about ANSI.SYS can be found in Appendix E.

Using the Cursor-Movement Commands

When DOS invokes the CLS command, the image on-screen is erased, and the cursor character and the prompt appear in row 1, column 1 of the upper left corner of the screen. Subsequent characters that appear on-screen are located one space farther to the right, until you reach the right side of the screen (and then the line wraps around) or until you press Enter to interrupt the line. Each new line begins one line farther down from the last.

The cursor also can be moved by using ANSI.SYS commands. The commands to move the cursor begin with the escape character (as described in the preceding section) followed by a left bracket. If you use EDLIN, this left bracket is in addition to the left bracket used to create the escape character (that is, you must put **Ctrl-V[[** in the EDLIN file). The instructions that follow show the escape-left bracket sequence as ESC[. If you use PROMPT to send commands to ANSI.SYS, type **$e[**; if you use EDLIN, type **Ctrl-V[[** whenever you see ESC[.

Reminder: *ANSI.SYS cursor-control commands begin with an escape character and [. If you use EDLIN, you must type [twice.*

Table 12.2 lists the ANSI.SYS commands you can use to move the cursor. Make sure that you type the commands with the capitalization given in the table. Notice that the variables *row* and *col* are used in place of the actual row or column number where you want to move the cursor; *lines* and *columns* are used in place of the number of rows or columns you want to move the cursor.

The ESC[s and ESC[u commands allow cursor movement without destroying the current location of the cursor. These two commands report the current position of the cursor, save it, and restore the cursor to that position. These commands are useful if you want to move the cursor to display something in a new location and then put the cursor back in its old position. For example, you could type the following PROMPT statement:

```
PROMPT $n$g$e[s$e[1;4H$d$e[u
```

This example changes the appearance of the system prompt by using cursor movement commands and two other PROMPT subcommands. $n and $g represent the current default drive and the greater-than symbol that make up the standard system prompt (such as C>). $e[s stores the current cursor position (don't worry where the information is stored, DOS takes care of that). $e[1;4H positions the cursor at row 1, column 4. The $d causes the system date to be printed at the cursor location (row 1, column 4). $e[u restores the cursor to its former position.

Table 12.2
ANSI.SYS Cursor-Movement Commands

Command	*Description*
ESC[*row*;*col***H**	Moves the cursor to the specified row and column. If you do not specify the row and column, the cursor moves to the home position. To move the cursor to row 3, column 8, for example, type the following: **ESC[3;8H**
ESC[*row*;*col***f**	Same as **ESC[***row*,*col***H**.
ESC[linesA	Moves the cursor up the specified number of lines. The column in which the cursor appears is unchanged.
ESC[linesB	Moves the cursor down the specified number of lines. The column in which the cursor appears is unchanged.
ESC[*columns***C**	Moves the cursor the specified number of columns to the right. If you do not specify the number of columns, the cursor moves one column to the right. This command is ignored if the cursor is already at the far right of the screen; in other words, you cannot wrap around to the next line.
ESC[*columns***D**	Moves the cursor the specified number of columns to the left. If you do not specify the number of columns, the cursor moves one column to the left. This command is ignored if the cursor is already at the far left of the screen; in other words, you cannot wrap around to the preceding line.
Esc[row;colR	Reports the current cursor position. The string is returned by ANSI console and is eight characters long.
Esc[2J	Erases the display and moves the cursor to the upper left corner of the screen.
Esc[K	Erases from the current cursor position to the end of the line.
ESC[s	Saves the current cursor position.
ESC[u	Restores the cursor to the location stored by **ESC[s**.

If you typed this PROMPT command at the DOS prompt (or included it in a batch file), whenever you pressed Enter, the standard prompt would appear on the appropriate line, but the system date would always print at the top of the screen (see fig. 12.1). DOS uses this prompt until you reboot the computer or reset the prompt (Chapter 6 describes the PROMPT command more fully).

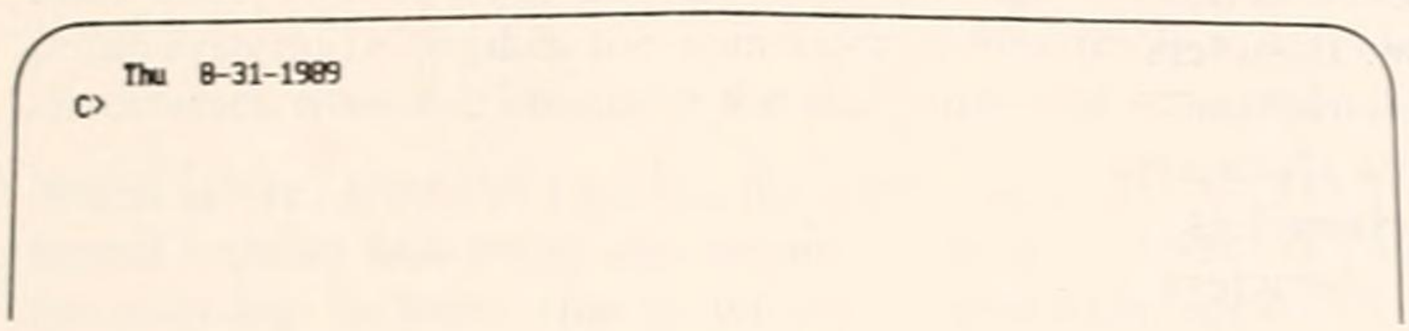


Fig. 12.1

The results of ***PROMPT ng$e[s$e[1;4Hde[u.***

Using the Screen Commands

Other ANSI commands are available to change the attributes of the video images displayed on your screen. If you have a color monitor, you can change the colors of the screen (or the prompt); if you have a monochrome monitor, you can alter the screen images to include reverse video, blinking, and other attributes.

Reminder: *Use screen commands to set the attributes of your monitor.*

The screen-attribute commands all begin with ESC[followed by a number or numbers and a lowercase *m*. You can include several of the numbers in a single command to set more than one attribute. If you string several numbers together, separate them with semicolons. The valid attribute numbers are shown in tables 12.3 and 12.4.

Table 12.3
ANSI.SYS Screen Commands for Monochrome Monitors

Number	*Result*
0	Normal characters
1	High-intensity characters
4	Underlined characters
5	Blinking characters
7	Reverse-video characters
8	Invisible characters

Reminder: *Screen-attribute commands begin with ESC[and end with m.*

If you use the PROMPT command, you can change colors using examples like the following:

```
PROMPT $e[0;1;31m
PROMPT $e[5;30;47m
```

If you use EDLIN, you can change colors with commands like these:

```
^V[[0;1;31
^V[[5;30;47
```

Table 12.4
ANSI.SYS Screen Commands for Color Monitors

Number	*Result*
30	Black characters
31	Red characters
32	Green characters
33	Yellow characters
34	Blue characters
35	Magenta characters
36	Cyan characters
37	White characters
40	Black background
41	Red background
42	Green background
43	Yellow background
44	Blue background
45	Magenta background
46	Cyan background
47	White background

The first examples produce high-intensity red characters, using the default background color. The second examples produce blinking black characters on a white background. Either of these commands is useful to incorporate into a system prompt.

Because you don't want to type long command sequences like these each time you boot DOS, you can incorporate the commands into a special batch file called AUTOEXEC.BAT. This file is explained in more detail in Chapter 13.

Reminder: *ECHO ON must be set for ANSI.SYS commands sent with PROMPT to work.*

For now, you may want to experiment with some batch files using ANSI.SYS. Following is a commonly used ANSI.SYS sequence that puts the date, time, and currently logged subdirectory in the upper right corner of your screen. ECHO ON must be set if ANSI.SYS is to interpret the PROMPT string, so don't include an ECHO OFF statement in your batch file. If you put this PROMPT line in a batch file that does other things, put an ECHO ON statement on the line preceding this one:

```
PROMPT $e[s$e[1;50H$t$h$h$h$h$h$h$d$e[u$n$g
```

This line creates the blinking prompt shown in figure 12.2.

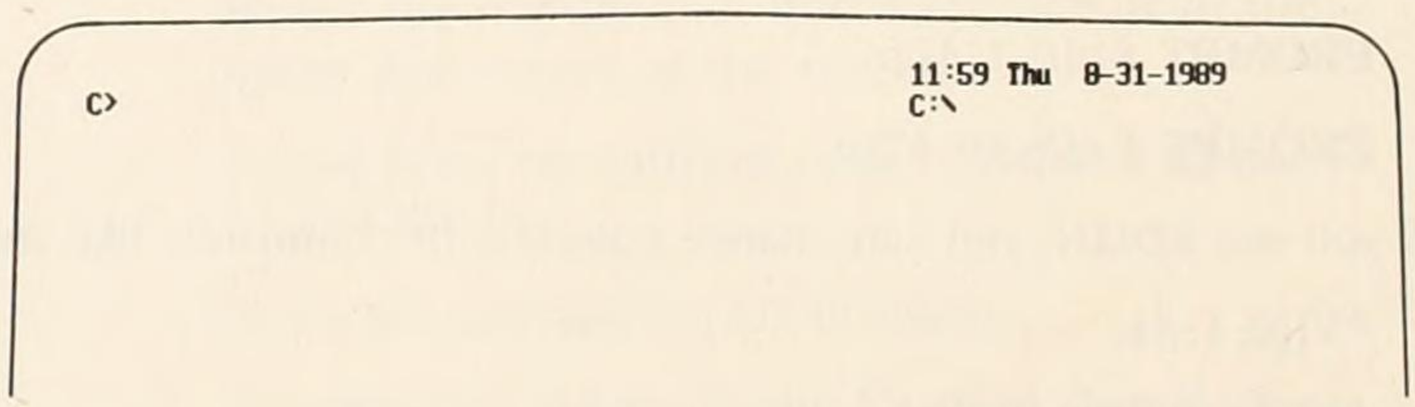


Fig. 12.2

The results of ***PROMPT $e[s$e[1;50H thhh$h hh$d $e[u$n$g.***

Using the Keyboard Commands

A lot happens when you press a key on your keyboard. Circuitry in your keyboard determines when a key has been pressed. The keyboard also can tell when a key has been released. Each time you press a key on the keyboard, a special code, called a *scan code*, is sent from the keyboard through the connecting cable to your computer system. DOS takes the scan code, compares it to a table of assignments, and determines what the character for that particular scan code is to be.

When active, ANSI.SYS handles the scan codes. If ANSI.SYS is not active, the DOS kernel handles this input and output. Ordinarily, ANSI.SYS passes the default value for each key to DOS. That is, when you press the uppercase letter A on the keyboard, ANSI.SYS receives a scan code of 65, and sends the DOS code for A to the operating system.

Understanding Scan Codes

Nearly every key on the keyboard has a unique scan code (the exceptions are explained shortly). For example, the uppercase letters A to Z are assigned scan codes 65 to 90. Lowercase letters a to z are represented by scan codes 97 to 122. Because these codes coincide with the ASCII codes for the same characters, the scan codes are easy to remember—if you have memorized the ASCII codes.

Reminder:
Scan codes are different from, but correspond to, the ASCII codes for the characters.

As you know, the uppercase letter A is produced by pressing the Shift key and the A key. Scan codes exist to represent other "shifted" states as well, such as Ctrl-A or Alt-A. Ctrl-A through Ctrl-Z are represented by scan codes 1 to 26, another easy-to-use set of numbers.

Scan codes are represented by 8-bit bytes, so only 256 scan codes are possible. Obviously, 256 numbers (0 to 255) aren't enough to represent all the possible key and shifted-key combinations. DOS gets around this limit by preceding some codes with a zero. Because no key is assigned a scan code of zero alone, adding a zero to the beginning of a scan code causes no confusion. When ANSI.SYS receives a zero scan code, it knows an *extended scan code* is being sent and interprets the next number it receives differently.

When ANSI.SYS receives a scan code of 65, for example, it knows the uppercase A has been pressed. When ANSI.SYS receives 0;65, it interprets that sequences as the F7 key. Several keys in the numeric keypad share the same scan code. The keypad zero and Shift-Ins keys, for example, have the same code: 48. If you notice, the zero is located on the Ins key, so the fact that they share a scan code should not be surprising. The Shift-Home, Shift-cursor left, Shift-cursor right, and other keys in that region share the same scan codes as the numbers on the keys. DOS checks whether the SHIFT or NUM LOCK key has been pressed to determine which value for the scan code is intended.

Reminder:
A few key combinations have no scan code at all.

A few key combinations have no scan code at all, such as Ctrl-cursor up and Alt-Home. These are null-key combinations that DOS cannot recognize.

Think of scan codes as values in tables that ANSI.SYS uses to look up the definitions it passes to DOS (see tables 12.5 and 12.6). You can change the definitions in the tables. When ANSI.SYS receives a scan code of 65 from the keyboard, it checks the table for the definition of that scan code. If you have not redefined the scan code, key 65 is the default value of A, and *A* is sent to DOS. If you substitute a new definition for scan code 65, such as *Hurrah for A!!*, that string is sent to DOS instead of the *A*.

Table 12.5
Scan Codes for IBM PC

Key	*Unshift*	*Shift*	*Control*	*Alt*
A	97	65	1	0;30
B	98	66	2	0;48
C	99	67	3	0;46
D	100	68	4	0;32
E	101	69	5	0;18
F	102	70	6	0;33
G	103	71	7	0;34
H	104	72	8	0;35
I	105	73	9	0;23
J	106	74	10	0;36
K	107	75	11	0;37
L	108	76	12	0;38
M	109	77	13	0;50
N	110	78	14	0;49
O	111	79	15	0;24
P	112	80	16	0;25
Q	113	81	17	0;16
R	114	82	18	0;19
S	115	83	19	0;31
T	116	84	20	0;20
U	117	85	21	0;22
V	118	86	22	0;47
W	119	87	23	0;17
X	120	88	24	0;45
Y	121	89	25	0;21
Z	122	90	26	0;44
1	49	33		0;120
2	50	64		0;121
3	51	35		0;122
4	52	36		0;123
5	53	37		0;124
6	54	94		0;125
7	55	38		0;126

Key	*Unshift*	*Shift*	*Control*	*Alt*
8	56	42		0;127
9	57	40		0;128
0	48	41		0;129
–	45	95		0;130
=	61	43		0;131
Tab	9	0;15		
Space	57			
Null	0;3			

Table 12.6
Extended Scan Codes for Numeric Keypad and Function Keys

Key	*Unshift*	*Shift*	*Control*	*Alt*
F1	0;59	0;84	0;94	0;104
F2	0;60	0;85	0;95	0;105
F3	0;61	0;86	0;96	0;106
F4	0;62	0;87	0;97	0;107
F5	0;63	0;88	0;98	0;108
F6	0;64	0;89	0;99	0;109
F7	0;65	0;90	0;100	0;110
F8	0;66	0;91	0;101	0;111
F9	0;67	0;92	0;102	0;112
F10	0;68	0;93	0;103	0;113
F11	0;133	0;135	0;137	0;139
F12	0;134	0;136	0;138	0;140
Home	0;71	55	0;119	
Crs-Up	0;72	56		
PgUp	0;73	57	0;132	
Crs-Lf	0;75	52	0;115	
Crs-Dn	0;77	54	0;116	
End	0;79	49	0;117	
PgDn	0;81	51	0;118	
Ins	0;82	48		
Del	0;83	46		
PrtSc		0;114		

You can assign the definition *Z* to the scan code for the letter A, if you want, or define the scan code for A as an entire sentence, such as *Hurrah for A!!*. In general, you do not often redefine keys you use with DOS (such as the normal alphanumeric keys). (If you redefine the A key as *Hurrah for A!!* and then press the keys for DIR A:, DOS understands what you type as *DIR Hurrah for A!!*:. This command is meaningless!) When you redefine keys, you do so for keys such as function keys not used for DOS or EDLIN editing, and Alt-key and Ctrl-key combinations not used by any program that uses ANSI.SYS for keyboard

Reminder:
Don't redefine keys you use with DOS or other programs.

interpretation. You can assign a different definition to each key combination, for example, so that you can have separate definitions for F10, Shift-F10, Alt-F10, and Ctrl-F10.

Caution: *Use only 200 bytes to redefine keys or your computer will lock up.*

The only limiting factor you have when defining scan codes is the amount of space DOS sets aside for these definitions. ANSI.SYS allows about 200 bytes for your new key definitions. If you use more than 200 bytes, the COMMAND.COM file is overwritten, and your computer locks up.

Redefining Scan Codes

To redefine the ANSI.SYS scan codes, you use escape sequences. You can use either the PROMPT command or EDLIN to create the line that communicates with ANSI.SYS. The syntax of the PROMPT command you use to redefine a scan code is as follows:

PROMPT $e[scan code;"new definition"p

If you use EDLIN, the following syntax works:

Cue: *To include a carriage return with the new key definition, use **13p** instead of **p** in the redefinition command.*

^V[[scan code;"new definition"p

If want the new key definition to end with a carriage return, substitute **13p** for **p** in the command.

In these commands, **scan code** is the scan code of the key you want to redefine (refer to tables 12.5 and 12.6). **new definition** is the new character or characters you want the scan code to return. If you want to define the scan code with a string of characters, enclose the string in quotation marks, as shown in the following example:

Reminder: *Enclose the new definition in quotation marks unless you are redefining the scan code with another scan code.*

PROMPT $e[65;"Hey I pressed A!"p

With EDLIN, the command is as follows:

^V[[65;"Hey I pressed A!"p

If you made this definition, every time you press Shift-A, ANSI.SYS sends the string *Hey I pressed A!* to DOS instead of the uppercase letter *A*.

If you want to define a scan code with another scan code (suppose that you want to substitute scan code 97 for 65 so that whenever you press Shift-A, a lowercase *a* appears), you can drop the quotation marks and simply use the second scan code, as shown in the following example:

PROMPT $e[65;97p

You could also make this same assignment with the following command:

PROMPT $e[65;"a"p

If you use EDLIN, the command is as follows:

^V[[65;97p

To return the 65 scan code to its original definition, you can either reboot the computer to return all scan codes to their default values or issue the following statement, putting the original scan code in the second position:

Cue:
To reset scan codes to their default values, reboot the computer.

PROMPT $e[65;65p

With EDLIN, the command is as follows:

^V[[65;65p

When you use extended scan codes, separate the zero and the second number of the scan code with a semicolon and separate the entire scan code from the new definition with another semicolon, as shown in the following examples:

PROMPT $e[0;59;"Hey I pressed F1!"p

^V[[0;59;"Hey I pressed F1!"p

The redefinition procedure is made a bit clearer with some examples. The scan codes 0;59 through 0;68 have the default definitions F1 through F10. You can change the definitions of these scan codes to produce *DIR A:<Enter>*, *DIR B:<Enter>*, and *DIR C: /P<Enter>* when you press F8, F9, or F10 with the following commands:

PROMPT $e[0;66;"DIR A:";13p
PROMPT $e[0;67;"DIR B:";13p
PROMPT $e[0;68;"DIR C: /P";13p

With EDLIN, the commands are as follows:

^V[[0;66;"DIR A:";13p

^V[[0;67;"DIR B:";13p

^V[[0;68;"DIR C: /P";13p

The values 0;66, 0;67, and 0;68 are the scan codes for F8, F9, and F10. The characters in quotation marks are the new definitions for these function keys. The definitions are followed by a carriage return as specified by the characters **13p**.

A number of programs enable you to redefine keys by creating macros that are even more sophisticated than these simple redefinitions. If you want to redefine keys quickly with short strings, however, the ANSI.SYS method is fun and easy to use. Best of all, it's free.

Summary

In this chapter you learned how to use ANSI.SYS to move the cursor, change keyboard definitions, and control the attributes that appear on-screen. Following are the key points to remember:

- DOS works with other devices through the use of installable driver files such as ANSI.SYS.
- ANSI.SYS extends the capabilities of ROM-BIOS and DOS for screen and keyboard control.
- You can communicate with ANSI.SYS by sending sequences of characters that start with the escape character.
- The escape character can be generated with the PROMPT command or EDLIN.
- You can control where the cursor moves on-screen with ANSI.SYS.
- You can set screen attributes with ANSI.SYS.
- Key redefinition involves the use of scan codes to change the strings DOS receives when a key is pressed.

Chapter 13 explains more about batch files and shows you how to integrate the things you have learned about ANSI.SYS into a batch file.

13

Learning Advanced Batch File Techniques

Chapter 11 discussed how batch files work and showed you some examples of how you can put them to work for you. This chapter explains batch file techniques in more detail.

This chapter discusses the following batch commands:

- IF
- FOR..IN..DO
- SHIFT
- CALL
- COMMAND /C

Reviewing Batch File Subcommands

Table 13.1 is a complete list of batch subcommands for DOS V3 through V4. You used many of these commands in Chapter 11. The commands not described in Chapter 11 are described in this chapter.

Any of subcommands from table 13.1 can be used in a batch file. Most of them can be used at the DOS system level as well. For example, you can type ECHO at the DOS prompt, with or without an argument. DOS either displays the string you type after ECHO or, if you provide no argument, tells you whether ECHO is ON or OFF. You can type PAUSE at the DOS prompt and DOS pauses. Neither of these commands is of much use at the operating-system level. The FOR..IN..DO command structure, however, can be quite useful at the operating-system level to carry out repetitive commands.

Reminder:
Most batch-file subcommands have limited use at the operating-system level.

Table 13.1
Batch File Subcommands for DOS V3 and V4

Subcommand	*Function*
@	Suppresses the display of a line on the screen (DOS V3.3 and later).
CALL	Runs another batch file and returns to the original batch file (DOS V3.3 and later).
CLS	Clears the screen and returns the cursor to the upper left corner of the screen.
COMMAND	Invokes a second copy of the command processor, COMMAND.COM, to call another batch file.
ECHO	Turns on or off the display of batch commands as they execute. Also can display a message on-screen.
FOR..IN..DO	Enables the use of the same batch command for several files.
GOTO	Jumps to the line following the specified label in a batch file.
IF	Enables conditional execution of a command.
PAUSE	Halts processing until a key is pressed. Displays the message `Press a key to continue. . .`
REM	Displays a message on-screen or allows for batch comments.
SHIFT	Shifts the command-line parameters one parameter to the left.

This chapter explains the four batch subcommands you didn't learn in Chapter 11: IF, FOR..IN..DO, SHIFT, and CALL. Examples of how each subcommand can be used are also provided.

Using IF

The IF subcommand is a "test and do" command. When the condition is true, IF executes the command that appears on the same line. When the condition is

false, IF ignores the line. The DOS IF subcommand works like the IF statement in BASIC. The subcommand can be used to test one of three conditions:

- The ERRORLEVEL of a program
- Whether a string is equal to another string
- Whether a file exists

Using IF To Test ERRORLEVEL

Reminder:
You can use the IF command to test the ERRORLEVEL of a program.

The first condition IF can be used to test is ERRORLEVEL. A better name for this condition would have been "exit level." ERRORLEVEL is a code some programs leave for DOS when they finish executing. In DOS V3.3 and later, only the BACKUP, GRAFTABL, KEYB, REPLACE, RESTORE, and XCOPY commands leave an exit code. Many other programs use exit codes, however.

An exit code of zero (0) usually means that everything is okay. Any number greater than 0 usually indicates that something went wrong when the program terminated: no files were found, the program encountered an error, or the operator aborted the program. Including an IF command in a batch file lets you test ERRORLEVEL to see whether a program worked properly.

If the exit code you test for is equal to or greater than the number you specify in the batch file, the ERRORLEVEL condition is true. You can think of this condition as a BASIC-like statement:

```
IF ERRORLEVEL > number THEN do this
```

The IF ERRORLEVEL command is most useful with any of the noncopyrighted public-domain utilities for user-keyboard input. These utilities have names such as INPUT.COM, ASK.COM, and the like. When your batch file uses these utilities, it can pause and wait for keyboard input. The utility puts a value in ERRORLEVEL related to the scan code of the key pressed. You can then make your batch file branch or perform some other task based on the key pressed. A batch file does not accept keyboard input except when the input is provided on a batch file command line.

Using IF To Compare Strings

Reminder:
You can use the IF command to test whether two strings are equal.

The second condition the IF command can be used to test is whether string 1 equals string 2. The IF *string1* == *string2* condition is normally used with parameters and variable markers. An example is a batch file you can create called ISIBM.BAT. This batch file contains the following single line:

```
IF %1 == IBM ECHO I'm an IBM computer
```

If you type **ISIBM IBM** at the DOS prompt, you see the following:

```
I'm an IBM computer
```

If you type something other than **IBM** as the first parameter, nothing appears on-screen.

If you don't provide enough parameters with the IF command, DOS replaces the parameter marker with nothing. Because DOS does not like to compare null parameters, it displays the message `Syntax error` and aborts the batch file.

To avoid the syntax-error problem, first test whether a parameter is empty before trying another test. You can see the general technique in this revised version of ISIBM.BAT:

```
IF "%1" == "" GOTO NOTHING
IF %1 == IBM GOTO IBM
ECHO What computer are you?
GOTO END
:IBM
ECHO I'm an IBM computer
GOTO END
:NOTHING
ECHO Pardon me, I could not hear you.
:END
```

Note: END is often used as a convenient GOTO label when none of the IF conditions are met. In this batch file, when the IF statements do not branch to NOTHING or IBM, the message `What computer are you?` appears, and the batch file branches to END. Because nothing follows the END label, the batch file quits.

Cue:
Use quotation marks to test whether a parameter has been provided for a variable marker.

The first line puts quotation marks around %1 and tests to see whether %1 is empty (is the same as a double set of quotation marks, or a null). If %1 is null (meaning ISIBM was called with no parameters), the line effectively reads as follows:

```
IF "" == "" GOTO NOTHING
```

This line traps the error condition of too few batch file parameters. If you need to check for a nonexistent parameter, use this format.

Cue:
You can use a period instead of quotation marks in the IF statement.

The quotation marks around the variable marker were used in this example because quotation marks are easy to understand, especially for BASIC programmers. In truth, a comparison with any letter, number, or symbol can do the job. One common procedure is to use a single period instead of the quotation marks, as shown in the following example:

```
IF %1. == . GOTO NOTHING
```

If no parameter is entered for %1, DOS interprets the line as follows:

```
IF . == . GOTO NOTHING
```

Use the syntax that is easiest for you to remember and understand.

If %1 equals nothing, DOS branches to the line following the label NOTHING and the message `Pardon me, I could not hear you` appears on-screen.

If %1 equals something other than nothing, DOS does not branch to NOTHING; instead, it executes the second line in the batch file. The second line of the revised ISIBM.BAT file tests to see whether **IBM** was entered. Notice that GOTO statements are used to jump around the parts of the batch file that should not be executed. If %1 equals IBM, the message `I'm an IBM computer` appears. If the parameter is not nothing, and is not IBM, the message `What computer are you?` appears.

Cue:
Use GOTO statements to jump around in the batch file.

Another fact to remember about the IF subcommand is that DOS compares strings literally. Uppercase characters are different than lowercase characters in the comparison. If you invoke ISIBM.BAT with **ISIBM ibm**, DOS compares the lowercase *ibm* to the uppercase *IBM* and decides that the two strings are not the same. The IF test fails, and `I'm an IBM computer` does not appear on the display.

Reminder:
The IF command compares strings literally.

Using IF To Look for Files

The third condition you can use the IF subcommand to test is IF EXIST *filename*. This IF condition tests whether the file *filename* is on-disk. If you want to test for a file on a drive other than the current drive, put the drive name in front of *filename* (for example, B:CHKDSK.COM).

Reminder:
You can use the IF command to test whether a file is on-disk.

You also can test for the opposite of these three IF conditions. That is, you can test whether a condition is false by adding the word NOT after IF. For example, you can use the statement IF NOT EXIST *filename* to check whether a file does not exist on the disk. If the result of the statement is true (the file does not exist), the command following the IF statement is executed.

Cue:
Use the IF command to test for the opposite of a condition by adding the word NOT.

Using FOR..IN..DO

FOR..IN..DO is an unusual and extremely powerful batch subcommand. The subcommand's syntax is as follows:

FOR %%variable IN (set) DO command

variable is a one-letter name that takes on the value of each item in *set*. The double percent sign in front of the variable is important. If you use a single

Reminder:
Make sure you use %% in front of the variable or DOS does not work properly.

percent sign, DOS confuses the symbol with a parameter marker and does not work properly.

set is the list of the items, commands, or disk files you want **variable** to take the value of. You can use wild-card file names with the set parameter. You also can use drive names and paths with any file names you specify. If you have more than one file name or other item in the set, use a space or comma between the names. **command** is any valid DOS command that you want to perform for each item in **set**.

Using a FOR..IN..DO Batch File

An interesting example of the use of FOR..IN..DO is a simple batch file that compares file names on a diskette with file names on another diskette or a hard-disk subdirectory. To see how this batch file works, copy some files to a diskette, either from another diskette or from a hard-disk subdirectory. Create the batch file CHECKIT.BAT and type the following in the file:

```
@ECHO OFF
IF "%1"=="" GOTO END
FOR %%a IN (B: C: D: E: b: c: d: e:) DO IF "%%a"=="%1" GOTO COMPARE
GOTO END
:COMPARE
%1
CD \%2
FOR %%a IN (*.*) DO IF EXIST A:%%a ECHO %%a is on this disk also.
:END
```

Put the diskette you want to compare in drive A: and type the following:

CHECKIT *drive*<*space*>*directory*

Notice that the directory you want to compare is optional, but if you specify it, you must separate it from the drive name with a space. This convention is followed because two batch file parameters (%1 and %2) are used to store the letter of the drive and the directory name. If you don't specify a directory name, the root directory of the drive you specify is used. If you type **CHECKIT B: \games**, for example, the file searches drive A: for the B:\games directory.

When the CHECKIT batch file is called, DOS first checks whether %1 is empty (if no parameters were provided when the command was issued). If %1 is empty, DOS branches to the end of the file. Otherwise, DOS goes to the third line and checks whether a valid drive letter was entered (the valid drive letters are B:, C:, D:, E:, b:, c:, d:, or e:). (If your system has more or fewer drives, change this list to reflect your configuration.) If an invalid drive letter is found, DOS branches to the end of the batch file. You must invoke this batch file with the drive name and the colon, separated by no space, or the comparison fails.

If the specified drive is valid, CHECKIT accesses the COMPARE section of the file. The first thing COMPARE does is to change the currently logged drive to the drive specified by %1. COMPARE then changes to the subdirectory specified by %2. If %2 is empty, the root directory of the disk is used. If %2 is a valid subdirectory name for that disk, the default directory is changed to %2. The CD\ command is valid even when %2 is null; no `Syntax error` appears in this instance. A `Syntax error` appears only when a null parameter makes the statement invalid.

The eighth line looks at all the files in the current directory to see whether a file with the same name as the one you specified when you invoked CHECKIT exists on drive A:. If a match is found, the name of the file, followed by the message `as on this disk also`, appears on-screen.

Using FOR..IN..DO at the DOS Prompt

You may find you want to issue commands like the ones in CHECKIT on an ad hoc basis at the DOS prompt. Instead of using a batch file for the example in the preceding section, you can manually change subdirectories and then type the FOR..IN..DO line (the line that does all the work in the batch file) at the DOS prompt. If you do use FOR..IN..DO outside a batch file, DOS requires that you enter only one of the percent signs.

Reminder:
If you use FOR..IN..DO at the DOS prompt, enter only one percent sign.

A useful way to use FOR..IN..DO is if you copy files from a diskette to the wrong subdirectory on a hard disk. You can then delete the files from the hard disk using FOR..IN..DO.

Using FOR..IN..DO with Wild Cards

FOR..IN..DO also enables you to do wild-card operations with certain commands that don't normally accept wild cards. The TYPE command, for example, does not accept wild-card file names; you must use a specific file name with the command. If you make a batch file, however, you can use FOR..IN..DO to type a series of files. If you created the batch file TYPER.BAT to do this operation, the file could consist of this statement:

```
@FOR %%d IN (%1 %2 %3 %4 %5) DO TYPE %%d
```

This statement substitutes up to five specific file names for the wild-card parameter you supply when you invoke TYPER.BAT and then executes the TYPE command for each file. If you type **TYPER *.BAT**, results similar to the following appear on-screen:

```
C>TYPE AUTOEXEC.BAT
   (AUTOEXEC.BAT's contents are displayed)

C>TYPE TYPER.BAT
   (TYPER.BAT's contents are displayed)

C>TYPE STARTUP.BAT
   (STARTUP.BAT's contents are displayed)

C>
```

Another example of the use of FOR..IN..DO is a little batch file that stops you from inadvertently formatting drive C:. In you want to use this batch file, you must place the DOS file FORMAT.COM in a subdirectory not on the PATH or in the current directory. When you create the batch file FORMAT.BAT, place it on the PATH or in the current directory. When FORMAT.BAT is created, type the following into the file:

```
@ECHO OFF
IF %1. == . GOTO WHATDRIVE
FOR %%d IN (A:,a:,B:,b:) DO IF %1 == %%d GOTO FORMAT
GOTO BADDRIVE
:FORMAT
C:\BIN\DISK\FORMAT %1 %2 %3 %4 %5
GOTO END
:BADDRIVE
ECHO You cannot format %1!
ECHO %1 is not a floppy disk drive
:WHATDRIVE
ECHO Please try again and specify the disk drive you
ECHO want to use, such as FORMAT A: or FORMAT B:
:END
```

The working line in FORMAT.BAT is the FOR..IN..DO command on line 3. This command tests whether the first parameter (which should be the disk-drive name) is A:, a:, B:, or b:. If a match is found, DOS jumps to the FORMAT part of the file. If no match is found, DOS jumps to the BADDRIVE part of the file. (Notice that if you type C:, no match is found; this file does not permit you to format drive C:.) Also notice that the second line of the file checks for a missing parameter.

If a matching drive name is found, DOS executes the FORMAT program for the drive specified by %1. (FORMAT.COM is located in \BIN\DISK, a subdirectory not on the PATH.) The additional variable markers (%2, %3, %4, and %5) are included in case you want to type some switches for FORMAT. If you don't type any switches, DOS fills these variable markers with nothing, and the blank variables do not interfere with FORMAT.

The error messages in this file are a little stingy: both error conditions (no parameter and the wrong drive name) use the same error message. If the specified drive is incorrect, the batch file goes to BADDRIVE and displays that message. The batch file then continues with the WHATDRIVE message.

Redirecting FOR..IN..DO

Cue:
DOS does not expand the FOR..IN..DO variable.

Before you continue your reading with the SHIFT subcommand, you should be familiar with an annoying aspect of the FOR..IN..DO subcommand and redirection. The FOR..IN..DO variable is not expanded by DOS. Like TYPE, MORE does not accept wild-card file names. (For that matter, MORE does not accept any file name. You must redirect the input of MORE.) You might think that a batch file like TYPER is perfect for MORE. To test this theory, include the following line in a batch file called MORER.BAT:

```
@FOR %%d IN (%1 %2 %3 %4 %5) DO MORE < %%d
```

Note: MORE is a DOS filter that displays a text file on-screen, one screen at a time, with the message `MORE` at the bottom of the screen. When you press the space bar, the next screen appears, with the message `MORE`, until the entire file has been displayed.

If you run MORER.BAT, the result is the `File not found` error message. Remove the @ from the batch file so that you can see the commands after DOS replaces the %1 through %5 parameters. Rerun the file, and the following appears on-screen:

```
C>MORER *.BAT
C>FOR %d IN (*.BAT    ) DO MORE < %d
File not found
C>
```

The spaces following *.BAT in parentheses show that DOS expanded %1 through %5. Instead of expanding %%d to the appropriate file names, DOS tries to open the file called %d because MORE's input comes from %d. You cannot use the FOR..IN..DO variable with redirection for either input or output. If you attempt to use >> %%d with a command within a batch file, the output of the command is placed in a file called %d. If you use %%d instead, the output from the command running is placed in %d. (Each time the %d symbol is encountered, the earlier version of %d is destroyed.) For more information about redirection, refer to your DOS manual.

The FOR..IN..DO command is versatile and allows one command to handle many files. However, for redirection of input and output, the command leaves something to be desired.

Using FOR..IN..DO with Other Commands

Reminder:
You can use FOR..IN..DO with other commands.

FOR..IN..DO works equally well with commands and file names. Instead of naming a set of files, you can name a series of commands you want DOS to carry out. Consider the following example:

```
FOR %%a IN (COPY ERASE) DO %%a C:*.*
```

In a batch file, this line first copies all the files on drive C: to the current directory and then erases the files from drive C:. You can put a replaceable parameter in the line instead of explicitly naming the drive and file specifications:

```
FOR %%a IN (COPY ERASE) DO %%a %1
```

To use this batch file, you must first change to the destination directory (for example, D:\BAK). When you invoke this second version of the batch file, you type the directory of files you want copied and removed. If you name the batch file MOVER.BAT, you can type the following to invoke the file:

```
MOVER C:\WP
```

MOVER.BAT copies all the files in the subdirectory C:\WP to D:\BAK and then erases the files in C:\WP. This file works much like the C&E.BAT file created in Chapter 11.

Moving Parameters with SHIFT

Cue:
Use SHIFT to trick DOS into accepting more than 10 parameters.

The SHIFT subcommand moves the parameters in the command one parameter to the left. SHIFT tricks DOS into accepting more than 10 parameters. The diagram of SHIFT is as follows:

```
%0 ← %1 ← %2 ← %3 ← %4 ← %5 . . .
  ↓
bit bucket
```

In this diagram, parameter 0 is dropped. The old parameter 1 becomes parameter 0. The old parameter 2 becomes parameter 1; parameter 3 becomes 2; parameter 4 becomes 3; and so on.

The SHIFTIT.BAT batch file is a simple example of the use of the SHIFT command:

```
:START
ECHO %0 %1 %2 %3 %4 %5 %6 %7 %8 %9
SHIFT
PAUSE
GOTO START
```

Suppose that you type the following:

SHIFTIT A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

The screen shows the following:

```
SHIFTIT A B C D E F G H I
Press a key to continue. . .
```

Notice that the batch-file name is displayed. This happens because %0 holds the name of the batch file before the first shift. Press a key to continue; ECHO now shows the following:

Reminder: The %0 variable marker holds the name of the batch file.

```
A B C D E F G H I J
Press a key to continue. . .
```

In this case, the file name has been dropped into the bit bucket. %0 now equals A. All the parameters have shifted one to the left. Press a key to continue; SHIFT continues moving down the list of parameters you typed initially. Press Ctrl-C when you want to stop.

SHIFT has many uses. You can use it to build a new version of the C&E.BAT file created in Chapter 11. The following modified version of the copy-and-erase batch file, called MOVE.BAT, shows a use for SHIFT:

```
:LOOP
COPY %1 C: /V
ERASE %1
SHIFT
IF NOT %1. == . GOTO LOOP
```

This batch file copies and erases the specified file or files. This batch file assumes nothing about the files to be copied; you can specify a disk drive, a path, and a file name. The batch file copies the files to the current directory on drive C: and then erase the files from the original disk or directory.

The last two lines shift the parameters to the left, test whether any parameters remain, and then repeat the operation if necessary.

Running Batch Files from Other Batch Files

On some occasions, you may want to run a batch file from another batch file. This section discusses three ways to run batch files from other batch files. One method is a one-way transfer of control. The other two ways show you how to run a second batch file and return control to the first batch file. These techniques are useful if you want to build menus with batch files or use one batch file to set up and start another batch file.

Calling a Batch File at the End of Another

Reminder: *Call a second batch file from the last line of the first batch file for a one-way transfer of control.*

The first method of calling a second batch file is simple. Include the root name of the second batch file as the final line of the first batch file. The first batch file runs the second batch file as if you had typed the second batch file's root name at the DOS prompt. To run BATCH2.BAT, for example, the final line of BATCH1.BAT is as follows:

```
BATCH2
```

DOS loads and executes the lines from BATCH2.BAT. Control passes in one direction: from the first batch file to the second. When BATCH2.BAT finishes executing, DOS displays the system prompt. You can consider this technique an interbatch-file GOTO. Control goes to the second file but doesn't come back to the first file.

Calling a Batch File with CALL

Reminder: *Call a second batch file and return to the first batch file using COMMAND /C (V3.0 through V3.2) or CALL (V3.3 and V4).*

In all versions of DOS, you can call a second batch file from the first, execute the second batch file, return to the first batch file, and continue processing the first batch file. With DOS V3.0 through V3.2, you use COMMAND, the DOS command processor. With DOS V3.3 and V4, you can use either the COMMAND or the CALL subcommand.

The syntax of the CALL subcommand is as follows:

CALL *d:path***filename** *parameters*

d:path\ is the optional disk drive and path name of the second batch file you want to execute. **filename** is the root name of the second batch file. You can place the CALL subcommand anywhere in the first batch file. DOS executes the batch file named by the CALL subcommand, returns to the first batch file, and executes the remainder of the first batch file. When you type the CALL subcommand, you can specify any *parameters* you want to pass to the batch file you're calling.

Understanding CALL Batch Files

The following three batch files demonstrate how CALL works. If you use DOS V3.3 or later, type these files into your computer. You can use uppercase or lowercase characters; uppercase characters are used here to highlight information. (If you use a version of DOS earlier than V3.3, do not type these files yet; some of the features used in these files don't work with versions before V3.3, as noted in the following discussion.)

```
BATCH1.BAT
   @ECHO OFF
   REM This file does the setup work for demonstrating
   REM the CALL subcommand or COMMAND /C
   ECHO This is the STARTUP batch file
   ECHO The command parameters are %%0-%0 %%1-%1
   CALL batch2 second
   ECHO CHKDSK from %0
   CHKDSK
   ECHO Done!

BATCH2.BAT
   ECHO This is the SECOND batch file
   ECHO The command parameters are %%0-%0 %%1-%1
   CALL batch3 third
   ECHO CHKDSK from %0
   CHKDSK

BATCH3.BAT
   ECHO This is the THIRD batch file
   ECHO The command parameters are %%0-%0 %%1-%1
   ECHO CHKDSK from %0
   CHKDSK
```

The first line of BATCH1.BAT sets ECHO OFF. With DOS V3 and V4, when you set ECHO OFF, it stays off. With DOS V2, ECHO turns back ON each time a new batch file starts. The next two lines in BATCH1 are REM comments. Because ECHO is OFF, the REM statements do not display when the batch file runs.

Reminder:
With DOS V3 and V4, ECHO OFF stays off for all the batch files.

The next two ECHO lines are similar for all three batch files. The first of the two lines identifies the batch file being used. The second of the two lines shows the zero parameter (the name by which the batch file was invoked) and the first parameter (the first argument) for the batch file. Notice that to display the strings *%0* and *%1*, you must use two percent signs (%%0 and %%1). If you use a single percent sign, DOS interprets the string as a replaceable parameter and does not display the actual symbol %.

Cue:
Use two percent signs in an ECHO statement if you want to display %.

The CALL statement in the first and second batch files invokes another batch file. In the first batch file, BATCH2.BAT is invoked. In the second batch file, BATCH3.BAT is invoked. In each case, a single argument is passed to the batch file being called: the word *second* to BATCH2.BAT and the word *third* to BATCH3.BAT.

Each batch file then displays its name (by using the %0 variable) as the file invoking the CHKDSK command, and then invokes CHKDSK. When DOS encounters the end of the batch file, DOS returns to the invoking batch file.

Analyzing Results of CALL Batch Files

Cue:
Use Ctrl-PrtSc to make the printer print whatever is echoed to the screen.

After you type the batch files explained in the preceding section, make sure that your printer is ready. Press Ctrl-PrtSc to turn on the printer. Type **BATCH1 FIRST**. When the printer is finished, toggle off the printer's capability to print echoed lines by pressing Ctrl-PrtSc again. If you do not have a printer to record the information displayed on-screen, press Ctrl-S to pause the screen as needed. When you have viewed the screen, press Ctrl-S again to "unpause" the screen display.

Check the printout or the screen display for the number of bytes of available RAM each time CHKDSK runs. Following is an abbreviated screen output from a PS/2 computer when **BATCH1 FIRST** is typed:

```
This is the STARTUP batch file
The command line parameters are %0-BATCH1 %1-FIRST
This is the SECOND batch file
The command line parameters are %0-batch2 %1-second
This is the THIRD batch file
The command line parameters are %0-batch3 %1-third

CHKDSK from batch3
Volume MODEL 60    created Apr 24, 1989 5:10p
  ...
  654336 bytes total memory
  453744 bytes free
CHKDSK from batch2
Volume MODEL 60    created Apr 24, 1989 5:10p
  ...
  654336 bytes total memory
  453840 bytes free
CHKDSK from batch1
Volume MODEL 60    created Apr 24, 1989 5:10p
  ...
  654336 bytes total memory
  453936 bytes free
DONE!
C>
```

Reminder:
DOS uses 96 bytes of memory each time you use the CALL command. You can run out of memory by nesting too many CALL commands.

Each time you use the CALL command, DOS temporarily uses 96 bytes of RAM until the called batch file finishes running. Because DOS uses free memory, you can run out of memory if you use many batch file CALL commands (although this is not likely). Not many people create long, "nested" calls. The accumulated memory-usage problem occurs only when batch files continue to call other batch files. A single batch file can use the CALL command as many times as desired and use only the same 96 bytes of RAM for each call.

You can use batch files like the ones described in this section with versions of DOS before V3.3 by making two changes. First, delete the @ character in the first line of BATCH1.BAT. The @ feature is not available in versions of DOS before V3.3. Second, change the reference to CALL in BATCH1.BAT and BATCH2.BAT to COMMAND /C. You then can run the batch files.

Calling a Batch File with COMMAND /C

If you use a version of DOS before V3.3, you can use the command interpreter to call other batch files. If you use V3.3 or later versions, you can use either CALL or COMMAND to call other batch files. The syntax of the COMMAND subcommand is as follows:

Reminder:
Use COMMAND /C with DOS versions before V3.3 to call other batch files.

COMMAND /C *d:path\filename* **string**

One difference between the syntax of CALL and COMMAND is that you *must* use COMMAND's /C switch. The /C switch accepts a single string (*d:path\filename* **string**) that can include spaces and any other information. COMMAND.COM is a program that loads and executes **string**. When COMMAND.COM executes **string**, two copies of COMMAND.COM are in memory: the original and the one executing **string**. When **string** is finished executing, the second copy of COMMAND.COM leaves memory, and the original copy regains control.

If you use COMMAND /C in the example batch files of the preceding section, the results are almost identical to results when CALL is used. Each copy of COMMAND.COM, however, uses more memory than the CALL subcommand. The amount of memory used by COMMAND.COM varies among versions of DOS, but is less than 4K.

Caution:
Do not make changes to the system environment when you use COMMAND /C because the changes are only temporary.

Another difference between CALL and COMMAND /C is worth noting. Because COMMAND loads a new version of COMMAND.COM, any changes you make to the environment are lost when you return to the batch file that invoked COMMAND. If you invoke another batch file with CALL and then change the environment, the changes are made to the copy of the environment currently used by that batch file. This distinction can be important if you write batch files that make changes to the environment as a way of "flagging" processes you carry out. With COMMAND /C, the current batch file does not know what happened; any batch files you run later also do not inherit the changed environment. If you use CALL, the environment changes are "permanent" and are available to the current batch file later on, as well as to other programs run by the same command processor.

To understand this process better, look at the following example. Because these batch files use the CALL command, they work with DOS V3.3 or later versions.

```
TEST1.BAT
   ECHO OFF
   SET VAR1=VAR1
   COMMAND /C TEST2
   CALL TEST3

TEST2.BAT
   SET VAR2=VAR2

TEST3.BAT
   SET VAR3=VAR3
```

Run the files by typing **TEST1** and pressing Enter. When the files are finished running, type **SET** (the DOS command that displays the environment) to see something like the following:

```
PATH=C:\BATCH;C:\MS-DOS;C:%\UTILS;C:\
COMSPEC=C:\MS-DOS\COMMAND.COM
VAR1=VAR1
VAR3=VAR3
```

Look at the last two lines. If the SET command lists the variables currently in memory, what happened to VAR2? It was set when TEST1.BAT used COMMAND /C to invoke TEST2.BAT. Because DOS loaded a new copy of COMMAND.COM to run TEST2.BAT, however, the variable VAR2 was lost when DOS returned to the original copy of COMMAND.COM to continue processing TEST1.BAT. Because CALL was used to invoke TEST3.BAT, however, VAR3 is not lost when DOS returns to TEST1.BAT.

Using the Special Batch File AUTOEXEC.BAT

Reminder: *DOS executes AUTOEXEC.BAT whenever the computer is booted.*

AUTOEXEC.BAT is a special batch file that DOS executes whenever the computer is booted, if the file is located in the root directory of your boot disk. You can include in AUTOEXEC.BAT a list of commands you want DOS to carry out at the beginning of each session.

Remember the following when you use AUTOEXEC.BAT:

1. The contents of AUTOEXEC.BAT must conform to the rules for creating batch files.
2. The date and time are not requested automatically when AUTOEXEC.BAT is executed after DOS is booted. To get the current date and time, put the DATE and TIME commands into AUTOEXEC.BAT. (Because PCs and XTs do not have a battery-backed

clock, you must manually reset the clock each time you start up the computer. Including the DATE and TIME commands in AUTOEXEC.BAT is a good idea for these computers.)

Viewing a Sample AUTOEXEC.BAT File

To better understand AUTOEXEC.BAT, look at the following example of a typical AUTOEXEC.BAT file. The line numbers are used only for reference.

Caution: Do not use the following file as your AUTOEXEC.BAT file. The file is given for explanatory purposes only and probably will not work on your computer.

```
(1) @ECHO ON
(2) PROMPT $e[32m
(3) PROMPT $e[s$e[2;53H$p$e[1;53H$t$h$h$h$h$h$h $d$e[u$n$g
(4) @ECHO OFF
(5) SET COMSPEC=C:\MS-DOS3\COMMAND.COM
(6) C:\KWIK\SUPERPCK /EM
(7) CLS
(8) PATH H:\;C:\BATCH;C:\MS-DOS3;C:\UTILS;C:\WIN386;C:\;
(9) C:\UTILS\FASTATKB 00 NUL
(10) \MOUSE\PCMOUSE2 /A0
(11) ECHO AT S0=0COM1:
(12) CALL C:\BATCH\RAMDISK
(13) CALL C:\BATCH\KEYWORKS
(14) CALL C:\BATCH\SIDEKICK
(15) SUBST E: D:\EDRIVE
(16) SUBST F: D:\FDRIVE
(17) SUBST G: D:\GDRIVE
(18) MENU
```

Understanding the Sample AUTOEXEC.BAT File

This section provides a line-by-line description of the AUTOEXEC.BAT file listed in the preceding section.

Because ECHO ON must be set for the PROMPT commands in the lines that follow, line 1 turns on echoing of screen commands. DOS normally begins batch files with ECHO ON. Setting ECHO ON here ensures that ECHO is ON, even in the unlikely event that AUTOEXEC.BAT is called by another batch file.

Reminder: *DOS normally begins batch files with ECHO set to ON.*

The PROMPT $e[32m statement in line 2 sets to green the screen characters of the color monitor used with this particular computer system.

The PROMPT $e[s$e[2;53Hpe[1;53Hthhhhh$h de[ung statement in line 3 customizes the system prompt with the time, date, and current directory name (refer to Chapter 12 for an example of how this command works).

ECHO ON had to be set for the PROMPT lines but is no longer required, so ECHO is turned OFF in line 4.

If you use several operating systems on your computer, such as OS/2, DOS V3.3, and V4, you can boot only one from the hard disk. The others boot from a diskette. The SET COMSPEC=C:\MS-DOS3\COMMAND.COM statement in line 5 boots DOS V3.3 from drive C:. This statement should be different for each AUTOEXEC.BAT file from which you boot. The purpose of this statement is to tell DOS or OS/2 where to look for its command interpreter.

The C:\KWIK\SUPERPCK /EM statement in line 6 loads a disk-caching program (disk-caching programs are explained in Chapter 14).

Because the screen may get sort of messy at this point, the CLS command in line 7 clears it.

The PATH H:\;C:\BATCH;C:\MS-DOS3;C:\UTILS;C:\WIN386;C:\; statement in line 8 sets up the PATH for this version of DOS. All DOS programs in this system, such as FORMAT, are kept in separate subdirectories named MS-DOS3, MS-DOS4, and so on. This PATH command ensures that each version of DOS accesses only its own files. This line varies depending on the AUTOEXEC.BAT file you're booting from (see line 5).

Line 9, C:\UTILS\FASTATKB 00 NUL, loads an optional, public-domain keyboard speedup utility.

The \MOUSE\PCMOUSE2 /A0 statement in line 10 loads the software for a particular mouse. Some systems put the mouse driver file in the CONFIG.SYS file instead.

The modem connected to this system and initialized with this AUTOEXEC.BAT file is connected to COM1. The modem is used for dialing calls with the SideKick program. Line 11, ECHO AT S0=0COM1:, tells the modem not to answer the phone.

This computer system has a RAM disk installed as drive H:. The CALL C:\BATCH\RAMDISK statement in line 12 invokes a batch file called RAMDISK to copy frequently used files to drive H: (RAM disks are described in Chapter 14).

Line 13, CALL C:\BATCH\KEYWORKS, calls a batch file that loads Keyworks, a keyboard-macro program. The batch file uses a utility to "mark" the memory

the program uses so that Keyworks can be uninstalled when using memory-hungry software like Ventura Publisher.

SideKick is another terminate-and-stay-resident (TSR) program used on this computer system. SideKick is loaded and unloaded in the same way as Keyworks. Line 14, CALL C:\BATCH\SIDEKICK, loads SideKick.

Line 15 (SUBST E: D:\EDRIVE), line 16 (SUBST F: D:\FDRIVE), and line 17 (SUBST G: D:\GDRIVE) have the same function. They enable a computer with a single 80M hard disk (divided into drives C: and D:) to mimic a computer that has both a 44M and an 80M hard disk (divided into drives E:, F:, and G:). By putting software from drive E:, F:, or G: on drive C: or D:, you can use the same batch files and other techniques on both computers. DOS "thinks" you have the same directory structure on both machines.

The last line in this batch file, MENU, calls an all-purpose menu batch file. Menus are described in the following section.

Building Menus

Cue:
Use DOS menus to make it easy to start programs from your computer.

Self-written menus like the one in this section are a convenient way to load individual programs without resorting to a complex shell. Following is a sample menu batch file, called MENU.BAT, you can customize and enhance for your own use. Once you understand how a basic menu works, you can jazz up the menu with ANSI.SYS commands, as described in the last section of this chapter.

```
@ECHO OFF
CLS
ECHO    Enter the following to run these programs:
ECHO        1.) Word Processing
ECHO        2.) Spreadsheet
ECHO        3.) DataBase Program
ECHO        4.) BASIC
ECHO        5.) Communications
ECHO    — ENTER CHOICE AND PRESS ENTER —
```

Reminder:
If you use DOS V3.0 or later, your menu batch files can specify drives and directories.

Simply substitute your own favorite programs for those in the sample batch file. Then create batch files that call up those programs and name these batch files 1.BAT, 2.BAT, 3.BAT, and so on, to correspond with the menu entries. If you use DOS V3.0 or later, your batch files can call programs by using path names. For example, the following 1.BAT file calls the WordStar program from the C:\PROGS directory:

```
@ECHO OFF
CALL C:\PROGS\WS
MENU
```

When you type **1** at the menu displayed by MENU.BAT, 1.BAT is invoked to access the WordStar program. The WS file can be a batch file that logs to the correct directory and then returns to the root directory when finished. Put the directory-change commands in WS.BAT instead of in 1.BAT so that you can substitute different versions of WS.BAT depending on what you want to do.

Cue: Create dummy batch files for the menu to display error messages when the wrong key is pressed.

If you want, you can create "dummy" batch files with the names of other numbers that will be called if the user of the menu types an incorrect number. These batch files can display error messages. For example, you can create the file 6.BAT to work with the sample MENU.BAT file:

```
ECHO OFF
ECHO No such menu choice! Try again!
PAUSE
CLS
MENU
```

If you type **6** from the menu, the message `No such menu choice! Try again!` appears and the PAUSE message `Press a key to continue. . .` appears. The file waits for you to press a key, clears the screen, and calls MENU.BAT to display the menu again.

Dressing Up the Menu with ANSI.SYS

Cue: *You can use ANSI.SYS techniques to enhance a menu.*

Now that you understand the basic concept behind MENU.BAT, you can spice it up with a few of the ANSI.SYS techniques you learned in Chapter 12. Examine the following new menu batch file. It may be a bit hard to decipher because of the escape sequences embedded in it. This listing shows the ^V[[escape codes you would see if you entered the batch file with EDLIN. If you use another editor that enables you to enter escape codes, replace ^V[[with the appropriate escape sequence and a left bracket ([).

```
@ECHO OFF
CLS
ECHO ^V[[31m^V[[1mEnter the following to run these programs:
ECHO      ^V[[1m^V[[32m1.)   ^V[[0m^V[[32mWord Processing
ECHO      ^V[[1m^V[[33m2.)   ^V[[0m^V[[33mSpreadsheet
ECHO      ^V[[1m^V[[34m3.)   ^V[[0m^V[[34mDataBase Program
ECHO      ^V[[1m^V[[35m4.)   ^V[[0m^V[[35mBASIC
ECHO      ^V[[1m^V[[36m5.)   ^V[[0m^V[[36mCommunications
ECHO ^V[[7m^V[[5m— TYPE CHOICE AND PRESS ENTER —^V[[0m
```

To use this batch file, load ANSI.SYS from your CONFIG.SYS file. For best results, use a color monitor. You can still see the effects of this menu if you have a monochrome monitor, but only the reverse video and high intensity characters are shown differently on your screen.

The third line in this file uses the ESC[31m code to change the screen foreground display color to red. The ESC[1m code changes the text to high intensity. As a result, the message `Enter the following to run these programs:` is displayed in high-intensity red characters on a color monitor, or just high-intensity characters on a monochrome monitor.

The next five lines display the options available on the menu, each in a different color. The number choice next to the option is displayed in high intensity, and the choice itself is shown in regular intensity. Each line begins with an ESC[1m code (high intensity) followed by an ESC[3*x*m code. The *x* in ESC[3xm is replaced by a number from 2 to 6, representing the color green, yellow, blue, magenta, or cyan. Recall that 32 through 36 are the numbers that represent these colors when using the foreground-color ANSI.SYS command.

The colors are switched from high intensity to regular intensity with the ESC[0ESC[3x command that follows on each line.

Finally, the message – `TYPE CHOICE AND PRESS ENTER` – is displayed in reverse, blinking video with the ESC[7m and ESC[5m commands. Reverse, blinking characters draw attention to the message.

Summary

In this chapter you learned these things about batch files:

- Several advanced subcommands can be used in batch files: IF tests for a condition or for the opposite condition. SHIFT shifts command parameters to the left. FOR..IN..DO can repeat a batch-file command for one or more files or commands.
- The @ character suppresses the display of a single line from a batch file.
- COMMAND /C and CALL can be used to invoke a second batch file and return control of the computer to the first batch file.
- AUTOEXEC.BAT is a special batch file that DOS calls when booting.
- Creating menus with batch files is a particularly good way to use your batch file techniques.
- You can use ANSI.SYS directives to dress up a menu with color and different intensities.

Chapter 14 provides information about the CONFIG.SYS directives you can use to configure and customize your computer.

Part IV

Advancing Your DOS Capabilities

Configuring Your Computer
Understanding the International Features of DOS
Using the DOS Shell

14

Configuring Your Computer

A great deal of flexibility is built into DOS. The default configuration of DOS may be fine for millions of users, but if you need to make some changes to better suit your needs, or to fine-tune your computer system, DOS has the capability.

DOS's flexibility translates into many more choices to make. To make intelligent choices, you must understand exactly how DOS is configured. Luckily, DOS makes fine-tuning your computer relatively simple. You don't need to learn arcane techniques or undergo painful "sysgen" (system generation) procedures. Customizing DOS is as easy as collecting a few commands in a plain ASCII file and rebooting. You learn everything you need to know about configuring your computer in this chapter.

This chapter covers the following CONFIG.SYS directives:

- BREAK
- BUFFERS
- DEVICE
- FCBS
- FILES
- LASTDRIVE
- SHELL
- STACKS
- REM
- SWITCHES
- INSTALL

Understanding CONFIG.SYS

Perhaps you either don't have a CONFIG.SYS file or have never examined the CONFIG.SYS file put on your hard disk by the person who installed your computer. Perhaps you don't know anything more about CONFIG.SYS than that it has the BUFFERS = 15 and FILES = 10 lines someone said you needed. If you are to use CONFIG.SYS, you should understand more about it. This section provides an explanation.

Reminder: *CONFIG.SYS is an ASCII text file in the root directory of the boot disk.*

After DOS starts, but before the AUTOEXEC.BAT file is invoked, DOS looks in the root directory of the boot disk for a file called CONFIG.SYS. If DOS finds CONFIG.SYS, DOS attempts to carry out the commands in the file before proceeding with the AUTOEXEC.BAT file. You cannot abort this configuration file by pressing Ctrl-C or Ctrl-Break as you can with AUTOEXEC.BAT.

Like AUTOEXEC.BAT, CONFIG.SYS is a plain ASCII text file that lists on separate lines special commands, called *directives*. You use a fairly limited set of directives as legal commands in CONFIG.SYS. As you will see, however, you can accomplish quite a lot with those commands.

Reminder: *Create CONFIG.SYS using COPY CON, EDLIN, or a word processor.*

If you don't have a CONFIG.SYS file, you can create it with the COPY CON CONFIG.SYS command. You also can create or edit an existing file using the DOS EDLIN text editor or a word processor that can output plain ASCII text. (EDLIN is explained in Chapter 10).

The CONFIG.SYS file contains the directives that alter some of DOS's functions and features. Table 14.1 lists the directives you can use in your CONFIG.SYS file and describes the functions they control.

Table 14.1
CONFIG.SYS Directives

Directive	*Action*
BREAK	Determines when DOS recognizes the Ctrl-Break sequence
BUFFERS	Sets the number of file buffers DOS uses
COUNTRY	Sets country-dependent information
DEVICE	Allows different devices to be used with DOS
FCBS	Controls file-handling for DOS V1
FILES	Sets the number of files to be used at one time
INSTALL	Installs memory-resident programs
LASTDRIVE	Specifies the highest disk drive on the computer (DOS V4)

REM	Flags a line as a remarks line (DOS V4)
SHELL	Informs DOS what command processor should be used and where the processor is located
STACKS	Sets the number of stacks that DOS uses
SWITCHES	Disables extended keyboard functions (DOS V4)

This chapter discusses the CONFIG.SYS directives—for the most part, in the order they are listed in table 14.1. (Chapter 15 discusses the COUNTRY directive.) The directives you probably use most are FILES, BUFFERS, and DEVICE; you may want to pay special attention to the discussions of these directives.

Telling DOS When To Break

Early in your computing career you probably learned that Ctrl-Break and Ctrl-C are helpful—but not foolproof—panic buttons. Who hasn't pounded frantically on the keyboard using one of these sequences in an attempt to unfreeze a locked-up or bombed applications program or to cancel—probably too late—an unwanted action?

Only an "oh-no" second may have passed from the time you pressed the panic button until DOS responded, but you still had time to wonder what took so long for DOS to pay attention. The answer is that DOS is busy doing other things most of the time and only looks for Ctrl-Break at intervals. The BREAK directive (identical to the DOS BREAK command) tells DOS when to check for this key sequence. BREAK does not enable or disable this feature.

Reminder:
Use BREAK to tell DOS when to check for the Ctrl-Break key sequence.

If you set BREAK ON in CONFIG.SYS, DOS checks to see whether you have pressed Ctrl-Break whenever a program requests some activity from DOS (performs a DOS function call). If you set BREAK OFF, DOS checks for a Ctrl-Break only when DOS is working with the video display, keyboard, printer, or asynchronous serial adapters (the ports at the back of the computer).

If you use programs that do a lot of disk accessing but little keyboard or screen work, you may want to set BREAK ON. This setting enables you to break out of the program if something goes awry.

The syntax structures for the BREAK directive are as follows:

BREAK ON
BREAK OFF

Because the default setting for this directive is BREAK OFF, you do not have to specify the directive at all if you do not want to set BREAK ON.

Using BUFFERS To Increase Disk Performance

The BUFFERS directive tells DOS how many disk buffers to use. Before the introduction of disk-caching programs (which also keep disk information in memory), BUFFERS had the potential for the greatest impact on disk performance. This directive was changed significantly for DOS V4. The older version of BUFFERS is discussed first, followed by a discussion of DOS V4 enhancements.

Reminder: *RAM access is much faster than hard-disk access.*

You don't have to be a computer expert to know that RAM is a lot faster than disk memory. RAM access speed is rated in nanoseconds (one billionth of a second); hard disks boast of access to data in as little as 18 to 28 milliseconds (thousandths of a second.) To translate that into human terms, the difference is like someone saying, "I'll get that for you in a second," or "I'll find that for you in eleven and a half days."

Clearly, whenever you can fetch data from RAM instead of from the disk, you are way ahead. You see how this concept is taken to its near-ultimate in the discussion of IBMCACHE.SYS and SMARTDRV.SYS later in this chapter. For now, consider the improvements that can be made with the BUFFERS directive.

A *buffer* is a reserved area of RAM set up by DOS to store information read from disk. If you have configured your system with enough buffers, DOS frequently finds the information it needs in buffered memory. Instead of idling for a few hundred million cycles during a disk access, your microprocessor may be able to go back to work in as little as a thousand cycles if the information is in a buffer.

A single buffer is about 512 bytes long (plus 16 bytes for overhead). The BUFFERS directive controls how many disk buffers your computer uses. The syntax for this directive is as follows:

BUFFERS = nn

Reminder: *A buffer is 528 bytes long; you can specify up to 99 buffers with DOS versions before V4.*

In this syntax, **nn** is the number of disk buffers you want. With versions of DOS before V4, you can have any number of buffers from 1 to 99. If you do not give the BUFFERS directive, DOS uses a default value of from 2 to 15. The default number of buffers depends on the type of disk you have, the amount of memory in your computer, and the version of DOS you use. Table 14.2 lists the different buffer configurations.

You can use fewer BUFFERS than the default numbers shown in table 14.2. Unless you use a disk-caching program, however, you may find you want more, rather than fewer, buffers.

Table 14.2
Default Number of Disk BUFFERS

DOS Version	*Number of BUFFERS*	*Hardware*
DOS pre-V3.3	2	floppy-disk drives
	3	hard disk
DOS V3.3/V4	2	360K floppy-disk drive
	3	any other hard or floppy-disk drive
	5	more than 128K of RAM
	10	more than 256K of RAM
	15	more than 512K of RAM

Understanding How DOS Uses Disk Buffers

When DOS is asked to get information from a disk, DOS reads information in increments of whole sectors (512 bytes). Excess data not required from that sector is left in the buffer. If this data is needed later, DOS doesn't have to do another disk access to retrieve it. Similarly, DOS tries to reduce disk activity when it writes information to the disk. If less than a full sector is to be written to the disk, DOS accumulates the information in a disk buffer. When the buffer is full, DOS writes the information to the disk. This action is called *flushing the buffer*. To make sure that all pertinent information is placed into a file, DOS also flushes the buffers when a program "closes" a disk file.

Reminder: *DOS reads and writes to the disk in full sectors and uses buffers as a holding area.*

When a disk buffer becomes full or empty, DOS marks the buffer to indicate that the buffer has been used recently. When DOS needs to reuse the buffers for new information, it takes the buffer that hasn't been used for the longest time.

The net effect of DOS's use of buffers is a reduction in the number of disk accesses DOS performs, because DOS reads and writes only full sectors. By reusing the least-recently used buffers, DOS retains information more likely to be needed next. Your programs and DOS run faster.

Cue:
Set a higher number of buffers if you do a lot of random disk work.

If your program does much *random* disk work (reading and writing information in different parts of a file), you may want to specify a higher number of buffers. The more buffers you have, the better the chance that the information DOS wants is already in memory (the disk buffer).

Some programs do not benefit much from using disk buffers. If your program mainly does *sequential* reads and writes (reading or writing information from the start of the file straight through to the end), using disk buffers doesn't give you a large advantage.

Fine-Tuning with Buffers

The number of disk buffers you should have depends on the programs you run on your computer and how much memory you have. If your day-to-day use of the computer does not involve accounting or database work (both of which use files with small groups of data), the right number of buffers is from 10 to 20. If you do accounting or database work, you should have between 20 and 40 buffers (more, if you use expanded memory with DOS V4). If you use many subdirectories, using more disk buffers can increase your computer's performance.

Caution:
If you use too many buffers, you may not be able to use some programs and utilities.

Because each disk buffer takes 528 bytes of memory, every two disk buffers you use cost just over 1K of RAM. If you use many memory-hungry programs and terminate-and-stay-resident (TSR) utilities, you may need to use a minimal number of buffers simply to preserve precious RAM.

Even if memory is no problem, DOS has a "magic" range for disk buffers. DOS V3 and V4 slow down if you specify between 40 and 60 disk buffers. This happens because DOS spends more time searching the buffers for information than reading or writing to the disk. Depending on the programs you run, you may find that specifying more than 30 buffers causes the system to become sluggish.

Cue:
Change the number of buffers until you think you have the best performance.

If you use a floppy-disk system, start with 10 disk buffers; start with 20 buffers for hard-disk systems. Fine-tune the number by increasing or decreasing the number by 1 to 5 buffers every few hours or once a day. Reboot DOS and examine its performance. Keep doing this until you think you have the best performance. You don't need to be exact, but just get the general "feel" of the computer's performance with different numbers of buffers.

These rules of thumb go out the window if you use sophisticated programs with memory-management routines of their own. If you have a disk cache (explained shortly), for example, most of the work done by buffers is handled automatically—and more intelligently. Memory managers for 386 computers, such as Quarterdeck's QEMM-386, set up their own disk buffers. If you fall into either of these categories, set the CONFIG.SYS BUFFERS directive to a relatively low number, such as 2. This action enables DOS to read a whole sector or two into memory, but doesn't waste memory on buffers that really aren't needed.

Understanding DOS V4 BUFFERS Options

DOS V4 added important options to the BUFFERS directive. The numbers of buffers you can set with DOS V4 was increased to 10,000—if you have expanded memory. In addition, you can now use something called a "look-ahead" buffer to increase disk efficiency.

Reminder: *If you have expanded memory, DOS V4 enables you to use up to 10,000 buffers as well as 8 look-ahead buffers.*

DOS V4 introduced support for expanded memory, and BUFFERS is one of the features that can use it. If you add the /X switch to the BUFFERS directive in your CONFIG.SYS file, DOS uses expanded memory to store buffered information. Calculate the number of buffers you can fit in the memory you have (each buffer is 528 bytes).

Look-ahead buffers are special buffers that DOS uses to store sectors ahead of the sector requested by a DOS read operation. A look-ahead buffer requires 512 bytes of memory. You can specify from 0 to 8 look-ahead buffers. If you specify 5 look-ahead buffers, for example, then every time DOS reads a sector from your disk, it also reads the following 5 sectors. Reading the additional sectors is simple for DOS because the read/write head is already positioned and operating to read the first sector.

If your application or DOS needs one of the sectors already read, DOS finds the sectors in the look-ahead buffer and does not have to make another disk access. As you can see, this can be a powerful performance enhancer.

The syntax for the expanded-memory and look-ahead-buffer features is as follows:

BUFFERS = buffers,*look-ahead_buffers /X*

For example, the directive BUFFERS=1000,8 /X tells DOS to set aside 1,000 buffers and 8 look-ahead buffers, using expanded memory. If enough expanded memory is not available, the directive is ignored, and DOS reverts to the default buffer value. To be sure you have enough memory for the buffers you want to specify, use the DOS V4 MEM command. The default is 1 look-ahead buffer; if you want to use look-ahead buffers, a good value to start with is 8.

Installing Device Drivers with DEVICE

You learned about device drivers in the discussion of ANSI.SYS in Chapter 12. DOS is supplied with a wealth of other device drivers you can use to make better use of your current—and future—computer hardware equipment. To make DOS aware of the device drivers you want to use, include the DEVICE directive in the CONFIG.SYS file. The syntax for the DEVICE directive is as follows:

Reminder: *Device drivers are interfaces between the operating system and the device.*

DEVICE = *d:path***filename**.ext /switches*

Note: You do not have to include spaces on either side of the equal sign in this syntax.

In this syntax, *d:* is the disk drive where the device-driver file resides, *path*\ is the directory path to the device-driver file, and **filename**.*ext* is the name of the device-driver file. */switches* are any switches needed by the device-driver software.

You can load as many device drivers as you need, but you have to use a separate DEVICE line for each driver you install. As each device driver is loaded, DOS extends its control to that device. Remember that the device driver must be accessible when DOS starts. For convenience, floppy-disk users should place device-driver files in the root directory. Hard-disk users can make a special subdirectory called \DRIVERS or \SYS and put the driver files in this directory, out of the way of daily files. If you put the driver files in a separate subdirectory, add the directory path name to the device-driver file name, as in the following examples:

```
DEVICE = C:\DRIVERS\ANSI.SYS
DEVICE = C:\SYS\ANSI.SYS
```

The preceding sections in this chapter began showing you how to use memory efficiently with the BUFFERS directive. The following sections show you how to gain extra horsepower from your system with some device drivers. Some of these drivers are supplied with DOS; others are supplied with the mouse, printer, or other software.

Understanding Disk Caches

Strictly speaking, the IBMCACHE.SYS device-driver file is not part of DOS (IBMCACHE.SYS comes on the Reference Diskette for IBM PS/2 Models 50, 60, and 80). Some versions of DOS come with the SMARTDRV.SYS file; other versions of DOS don't have it; if your version of DOS doesn't come with the file, you have to purchase Windows to get SMARTDRV.SYS. No discussion of device drivers is complete without an understanding of disk-cache systems. The concepts discussed in this section apply to both of these programs as well as other caching programs.

Reminder:
A disk cache is an intelligent buffer.

You can think of IBMCACHE.SYS and SMARTDRV.SYS as large, smart buffers. Like a buffer, a disk cache operates from a section of RAM you request from DOS. Both a disk cache and a disk buffer accumulate information going to the disk or coming from the disk. When information is read from the disk, the information is placed in a buffer or cache; the information requested by the program is sent to the program. Like a buffer, a disk cache can read more information than was requested by the program. When information is written to the disk, the information may be stored in the buffer or cache until the buffer or cache is full, and then written to the disk. (Most caches have a "write-through" feature:

the cache writes information to disk frequently to avoid data loss if the system is turned off or loses power. Smart caches eliminate redundant writes by putting information on disk only when the data is different from what is there already.) If you use a cache or buffers, your program performance improves because slow disk accesses are replaced by higher-speed memory-to-memory transfers.

Although a DOS disk buffer is a form of a disk cache, a good caching program outperforms a buffer. Disk caches are better performers because they have more "intelligence" than buffers. You may find that the performance of a disk cache is hampered if you make the cache too large or too small. Test different applications you use and vary the size of the cache to find the optimum performance.

A disk cache remembers what sections of the disk have been used most frequently. When the cache must be recycled, the program keeps the more frequently used areas and discards those less frequently used. Buffers, as you recall, are recycled based on how old the buffer is, regardless of how frequently the buffer is used.

In an absolutely random disk access, where program and data files are scattered uniformly across the disk, the cache method of recycling the least frequently used area is more efficient than the buffer method of recycling the oldest area. A cache tends to keep in memory the areas of the disk that are most heavily used.

Like the DOS V4 BUFFERS directive, many disk caches have look-ahead capabilities, a technique in which the cache reads more sectors than the program requests.

Reminder:
Disk caches have look-ahead capabilities.

In many cases, using a disk cache to help with disk activity is preferable to using a RAM disk program like RAMDRIVE.SYS (discussed later in this chapter). A RAM disk helps only with files copied to the RAM disk (loading a program with overlays into a RAM disk may be better than using a cache to speed up a slow disk). The cache's activity is effortless and transparent to the user; a RAM disk requires separate directives to access the needed files.

A disk cache is also preferable to a RAM disk because the danger of losing information with a cache can be less. You have a greater probability of losing information when files are written to a RAM disk than when files are written to the cache. The cache, like a DOS buffer, holds the information for only a few seconds before writing it to the disk. During this brief period, the information is at risk. If power is lost to the RAM disk, however, all files on the disk are destroyed. You have to issue commands to copy the information in the RAM disk to the safety of a hard disk; the RAM disk does not do this automatically.

Cue:
A cache is safer than a RAM disk because information is held in memory for only a short time.

In two situations, however, a RAM disk is preferred to a disk cache. When you need to copy files between diskettes on a computer with a single floppy-disk drive, copying to and from the RAM disk is convenient. You first copy the files

to the RAM disk, change diskettes, and then copy the files from the RAM disk to the second diskette. A disk cache cannot help you copy files between diskettes.

You also may prefer to use a RAM disk if you frequently use nonchanging files that can be placed completely on the RAM disk. A good example is Turbo Lightning; you can place its dictionary and thesaurus files on a RAM disk. Because you access these files frequently—but rarely the same spots in the files—having the files on a RAM disk provides better performance than having the files in a disk cache.

Installing IBMCACHE.SYS

IBMCACHE.SYS is a file that accompanies any IBM PS/2 that uses extended memory (this excludes Models 25 and 30). IBMCACHE is a disk-caching program specifically made for PS/2 computers. IBMCACHE.SYS is found on the Reference Disk that comes with the computer. You must boot from the Reference Disk to install the cache the first time.

To install IBMCACHE.SYS the first time, you must run the program IBMCACHE from the Reference Diskette. Because the IBMCACHE.COM program and the IBMCACHE.SYS disk cache itself are hidden files on the diskette, you cannot copy the files to your computer using the DOS COPY command. You have to install the disk cache.

First, start up DOS on your PS/2 system. After DOS is running, insert the Reference Diskette into drive A: and type the following:

A:IBMCACHE

Your screen should look something like figure 14.1. The highlighted bar should be on the first option, `Install disk cache onto drive C:`. When you press Enter to select the option, a window pops up giving the information shown in figure 14.2. Press Y to install the cache. The pop-up window goes away, the floppy-disk and hard-disk lights go on, and a message that the cache is installed appears in another window (see fig. 14.3). Press Enter to continue.

If you have extended memory, move the highlighted bar on the Disk Cache main menu down to option 3, `Change disk cache settings`, by tapping the down-arrow key twice. Press Enter. The Change Disk Cache Settings screen appears, as shown in figure 14.4. The following sections explain how to use this screen.

Changing the Cache Location

Change the first option, `Cache Location`, from `Low Memory` to `Extended Memory` by pressing either the F5 or F6 function key.

```
Disk Cache Main Menu

 1. Install disk cache onto drive C:
 2. View disk cache settings
 3. Change disk cache settings
 4. Remove disk cache from drive C:

Use ↑ or ↓ to select. Then press Enter.
Esc=Quit   F1=Help
```

Fig. 14.1

Initial installation screen for IBMCACHE.

```
Disk Cache Main Menu

 Install Disk Cache                    Page  1 of  1

 The IBM disk cache will be installed
 onto fixed disk drive C.

 After installation, when the computer
 is started using IBM DOS on fixed disk
 drive C, the disk cache will start
 automatically.

 Install disk cache? (Y/N)
 Esc=Quit
```

Fig. 14.2

IBMCACHE installation information.

```
Disk Cache Main Menu

 1. Install disk cache onto drive C:
 2. View disk cache settings
 3. Change disk cache settings
 4. Remove disk cache from drive C:

Use ↑ or ↓ to select. Then p
Esc=Quit   F1=Help

 Information

 The IBM disk cache has been installed
 onto fixed disk drive C.

 Press Enter to continue.
```

Fig. 14.3

Successful installation of IBMCACHE.

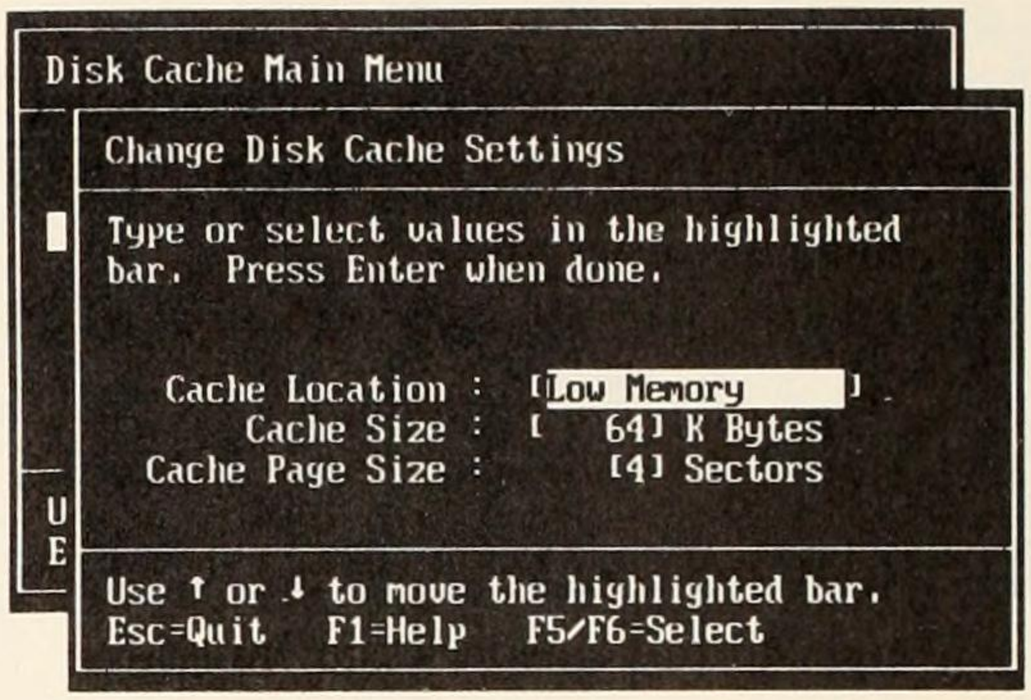


Fig. 14.4

The Change Disk Cache Settings screen.

Changing the Cache Size

Cue:
The initial size of the cache is 64K, but you can expand it to fill extended memory.

The initial size of the cache is 64K. A computer with 1M of memory has 384K of extended memory. You may want to expand the size of the cache to incorporate all the extended memory, or you may want to save some of the memory for a RAM disk. (The latest computer systems, particularly 386-based systems, may use some extended memory to "shadow" ROM code in faster RAM. If you have a VGA board or other peripherals, some ROM may be mapped into extended memory, which decreases the amount of extended memory that can be used as a cache.)

If you want to change the size of the cache, press the down-arrow key on the Change Disk Cache Settings screen to move to the `Cache Size` option. Press F5 to reduce the cache size; press F6 to increase the size. Try pressing the F5 and F6 keys to see some of the possible sizes for the cache. Alternatively, you can type the desired cache size.

If you don't use a RAM disk in extended memory, use all the extended memory for the cache by specifying a cache size of 384K. If you use a RAM disk in extended memory, use the leftover memory for the cache: Subtract the size, in K, of the RAM disk from 384K and use the resulting figure as the size of the cache.

You may notice that the size choices offered when you press F5 or F6 do not match your calculations. If you do not see a number that matches your calculations, simply type the number you want.

Changing the Cache Page Size

Cue:
You can request that DOS read more than the required number of sectors for the disk cache.

The final option on the Change Disk Cache Settings screen is `Cache Page Size`. This option specifies the number of sectors to be read whenever DOS reads any information from the disk. Regardless of how many sectors a program requests that DOS read, DOS always reads the number of sectors you specify here (in this respect, the page size is similar to a look-ahead buffer). DOS

always reads the minimum amount of sectors required; if you specify a page size larger than what is required, more sectors are read. This strategy is excellent for programs that start at the beginning of a file and read to the end, but may not be good for programs that read files randomly.

The starting value for `Cache Page Size` is 4 sectors. The values 2, 4, and 8 are valid for this option. Do not change this value initially. No single optimum value can be given for this option; you and your programs must determine the best value. Run your system for several hours or days. If you find the performance of some of your programs declining slightly, rerun IBMCACHE and change this value to 2. You reduce the number of sectors to be read because your programs are reading files randomly rather than sequentially; DOS is wasting time reading sectors that are not used by the program.

Once you have finished changing disk-cache options, press Enter. You see a message stating that the changes you made have been saved (see fig. 14.5). Press Enter again to return to the DOS prompt.

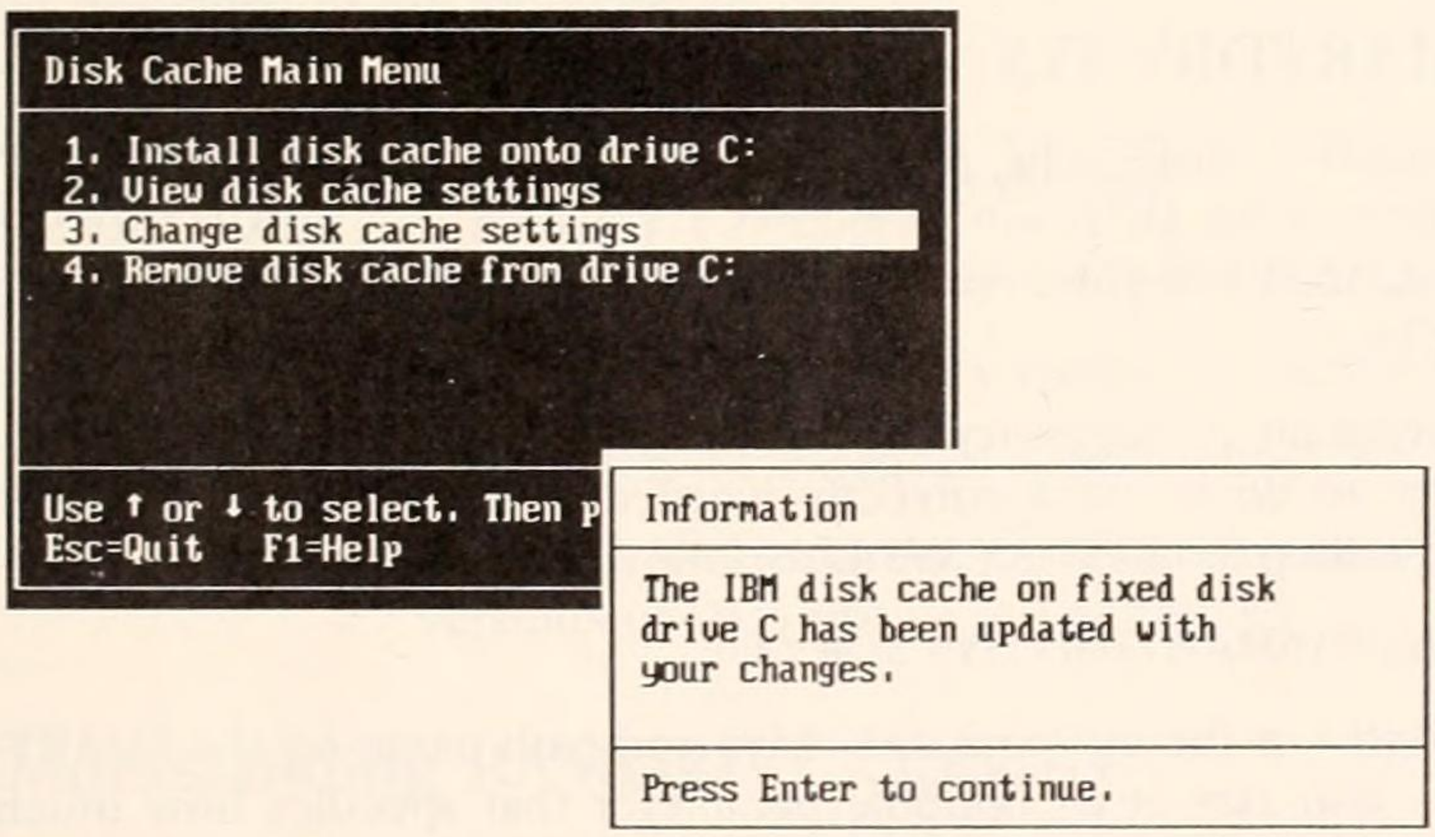


Fig. 14.5

Successful change of IBMCACHE options.

To start the cache program, restart DOS by pressing Ctrl-Alt-Del; you soon should see the following message:

```
Disk Cache Version 1.0

Copyright 1987 by IBM Corp.
Allocating Cache Buffers
Cache Initialization Complete
```

The cache has been successfully installed. Whenever you boot DOS, the cache is reinstalled.

Changing the IBMCACHE Settings in CONFIG.SYS

Reminder:
Change disk-cache settings by editing CONFIG.SYS.

If you want to change options after the cache has been installed, you can either rerun IBMCACHE or edit the IBMCACHE line in your CONFIG.SYS file (when you installed the cache, the CONFIG.SYS file was altered to include this line). The syntax for the IBMCACHE.SYS program is as follows:

DEVICE = *d:path***IBMCACHE.SYS** *cachesize /E or /NE /Psectors*

In this syntax, *d:path*\ is the optional disk drive and path name for the IBMCACHE.SYS program, and *cachesize* is the size of the cache in kilobytes.

The */E* and */NE* switches designate where the cache is to be placed. /E designates extended memory, and /NE designates the 640K of low memory. You can use either switch, but not both.

/Psectors designates the number of disk sectors that DOS should read at once. *sectors* can be set to 2, 4, or 8. Notice that, unlike many DOS switches, */Psectors* requires no colon to separate the P from the number of sectors.

Installing SMARTDRV.SYS

With MS-DOS comes a disk-cache program called SMARTDRV.SYS. (SMARTDRV.SYS also comes with Microsoft Windows.) You can use SMARTDRV.SYS instead of IBMCACHE.SYS, if you want. This section explains how to install and use SMARTDRV.SYS.

No installation program is necessary for SMARTDRV.SYS. To install this program, all you have to do is put a correctly worded directive in CONFIG.SYS. The syntax for installing SMARTDRV.SYS is as follows:

DEVICE = *d:path***SMARTDRV.SYS** *size /A*

Cue:
If you do not have expanded or extended memory, you cannot use the SMARTDRV.SYS program.

In this syntax, *d:path*\ is the optional disk drive and path name for the SMARTDRV.SYS program, and *size* is an optional parameter that specifies how much memory you want SMARTDRV to use. The default size is 256K. If you don't have expanded or extended memory, you may want to specify a smaller size than 256K to limit the cache size. Unless you limit the size of the cache, you may not have enough memory to run large programs.

The */A* parameter tells SMARTDRV to use expanded memory, if it is available, or to use extended memory as expanded memory. If this switch is not used, SMARTDRV uses extended memory.

After you insert the SMARTDRV.SYS directive in CONFIG.SYS and boot the computer, you see something like the following message:

```
Microsoft SMARTDrive RAM Cache version x.xx
Cache Size: 256K in 80286 Extended Memory
```

The version number and size of the cache vary depending on how SMARTDRV was set up.

Using RAMDRIVE To Make a RAM Disk

RAM disks are given a drive allocation such as D: or E:. You can use RAM disks just like an ordinary drive, but you lose the contents of the drive when you turn off the computer.

Another device driver you can install is RAMDRIVE.SYS (also called VDISK.SYS in some implementations of DOS). This program enables you to install an imaginary disk in memory. Like buffers and disk caches, a RAM disk is much faster than ordinary disk access.

To install RAMDRIVE and create an imaginary (or virtual) disk, just insert a line in CONFIG.SYS. The syntax to invoke RAMDRIVE from CONFIG.SYS is as follows:

DEVICE = *d:path***RAMDRIVE.SYS** *bbbb sss ddd /E:max or /X*

In this syntax, *d:* is the disk drive and *path*\ is the directory path for RAMDRIVE.SYS. The options for RAMDRIVE are listed in the following chart and described in the following section:

Option	*Description*
bbbb	The size (in K) of the RAM disk
sss	The size (in bytes) of the RAM-disk sectors
ddd	The number of directory entries (files) for the RAM disk
/E:max	The switch for extended memory use
/X	The DOS V4 switch for expanded memory use (some versions of DOS use /A instead of /X)

Understanding RAMDRIVE Options

This section describes the parameters you can use with the RAMDRIVE.SYS program. The arrangement of the options is shown in the syntax line in the preceding section.

The RAMDRIVE *bbbb* option can range from 1K to the maximum amount of memory available minus the 64K that DOS requires for other programs. For most PCs and compatibles, the maximum amount of memory is 640K. ATs and compatibles can have up to 16M of RAM, so you can set up more than one RAMDRIVE. For example, you can conceivably set up one 3M and three 4M virtual disks (*virtual disk* is another name for RAM drive).

Cue: If you have an AT-type computer with extended memory, you can set up one or more RAM disks in extended memory.

The default value of *bbbb* is 64K. With DOS V3.2, V3.3, and V4, RAMDRIVE automatically reduces the RAM disk's size if you forgot to leave 64K of memory free for your programs. With DOS V3.0 and V3.1, if you specify a value for *bbbb*

larger than the amount of your computer's RAM, RAMDRIVE uses the default of 64K. If you have less than 64K of memory available, RAMDRIVE displays an error message and does not install.

The *sss* option is the size of the sectors for RAMDRIVE. You can specify one of three sector sizes: 128, 256, or 512 bytes. The default sector size is 128 bytes. (You are allowed to specify the sector size for a RAM disk; DOS disks usually use a sector size of 512 bytes.) Set *sss* to 512 if you store larger files on the RAM disk or if you want to maximize disk speed when you use the RAM disk. If you store many small files, set *sss* to 256 or 128 so that less space is wasted would be if you specified a larger sector size. If you set a smaller sector size, the disk must be read more often, but the space in the RAM disk is used more efficiently.

The *ddd* option is the number of directory entries for the RAM disk. Each file you store on the RAM disk uses one directory entry. Because the volume label that RAMDRIVE creates takes one of the directory entries, *ddd* actually specifies the number of files, minus one, that your RAM disk can hold. You can specify a value for *ddd* from 2 to 512 files; the default is 64. Set the number of directories based on the size of the RAM disk and the size of the files you are storing.

Cue:
Use the /E switch to place a 4M RAM disk in extended memory.

/E is the switch RAMDRIVE uses to access extended memory, that is, memory above the one-million-byte point. If you use the /E switch, RAMDRIVE uses 768 bytes of regular memory but places the RAM disk itself in extended memory. If you use the /E switch, the maximum size of the RAM disk is 4M. You can use multiple RAM disks, limited only by the amount of available extended memory and DOS's regular memory. Because each RAMDRIVE uses 768 bytes of regular memory, specifying many RAM disks may reduce regular memory below 64K. If this happens, the last RAM disk you try to install does not install.

DOS V3.0 uses the /E switch alone; later versions add the *max* parameter to the switch. *max* is the maximum number of sectors that can be transferred at one time from the RAM disk. Certain programs, primarily communications programs, may lose characters if the CPU's attention is directed to the RAM disk for too long. If you have such a problem, set *max* to 7 or less. Otherwise, omit the parameter or specify /E:8.

If you have enough extended memory, you can set up several RAM disks. Just specify additional DEVICE=RAMDRIVE statements in CONFIG.SYS. Each statement invokes a fresh copy of RAMDRIVE and installs additional RAM disks.

To use a RAM disk that has the features of a 360K, double-sided diskette, for example, use this command line:

```
DEVICE = RAMDRIVE.SYS 360 512 112
```

If you want a RAM disk that uses all the extended memory on a computer with 1M of RAM, use this command line:

```
DEVICE = RAMDRIVE.SYS 384 512 112 /E:8
```

This statement creates a RAM disk of 384K (384K is the difference between 1M of RAM and 640K of conventional memory), with 512-byte sectors and 112 directory entries. The /E:8 switch tells DOS to install the RAM disk in extended memory and transfer 8 sectors to and from the RAM disk at a time. If you omit the /E switch, DOS installs the RAM disk in the 640K of DOS memory.

If you use DOS V4, you can use the */X* parameter, which enables you to use expanded memory (instead of extended memory) for the RAM disk. To use this switch, you must load the expanded-memory driver (such as XMA2EMS.SYS) before loading RAMDRIVE.SYS. You cannot use both the /X and /E switches for the same RAM disk. You may create different RAM disks, however—some using extended memory and others using expanded memory—if you have enough memory available. Simply insert multiple DEVICE statements in CONFIG.SYS.

Cue:
If you have DOS V4, you can put RAM disks in expanded memory instead of extended memory.

Learning More about RAM Disks

The amount of RAM in your computer, the programs you use, and the convenience of a RAM disk play a part in determining the size of RAM disk you use—or whether you even should use a RAM disk. This section helps you determine whether you should bother using a RAM disk, and if so, what size may help you the most.

A RAM Disk in 640K of Memory

Although all computers can accommodate a RAM disk, your computer may not be able to accommodate a RAM disk while using other programs. When DOS V4.01 is loaded on a computer with 640K of memory, for example, 581K is left for programs to use (640K is actually 655,360 bytes). DOS V4.01, using the CONFIG.SYS FILES and BUFFERS directives and the ANSI.SYS device driver, uses about 74K of RAM.

Reminder:
Although any computer can accept a RAM disk, you may not have enough memory for a RAM disk and an active program.

If you use Lotus 1-2-3 Release 2.2 on this computer, you must have 223K of RAM for the program, plus enough memory left over for worksheets. Loading 1-2-3 leaves 358K for worksheet creation. If you create a RAM disk in the remaining 358K of conventional memory, you decrease the memory you can use for a worksheet.

As another example, dBASE IV uses 412K of RAM, leaving 169K of memory. Although dBASE IV does not use RAM for workspace in the same way that 1-2-3 does, you may limit some features of dBASE IV, such as running DOS commands from dBASE, if you create a RAM disk.

The bottom line is this: if you have only conventional memory, you may find that a RAM disk does not optimize your use of memory. Exceptions exist, of course. If the program you use does not take up a lot of memory, you may be able to use a RAM disk. You may want to limit the size of the RAM disk to no more than 100K, however.

Cue:
If you have 640K of RAM, you may want to specify a RAM disk of only 100K.

If you set up a 100K RAM disk, you cannot store a large quantity of information on the RAM disk. You probably would use the RAM disk only for temporary storage of rather small files. You may want to use the following directive in your CONFIG.SYS file:

```
DEVICE = C:\RAMDRIVE.SYS 100 128 64
```

This statement creates a 100K RAM disk, with 128-byte sectors and 64 directory entries. Because the RAM disk uses some of the allocated space for the directory and file-allocation table (FAT), only 98.9K is actually usable.

The small sector size of 128 bytes is assigned in this example to maximize the performance of the RAM disk. If you set up the disk with 128-byte sectors, and the disk contains 10 files of 600 bytes each, 92.5K is available for further storage. If you set up the RAM disk with 512-byte sectors and have the same files, only 89K would be available.

Cue:
Use smaller sector sizes for smaller files to save memory; use larger sector sizes for larger files to save time.

Because one file does not share a sector with another file, a single 600-byte file uses five 128-byte sectors, wasting 40 bytes. The same 600-byte file uses only two 512-byte sectors; however, 497 bytes are wasted. From this example, you can see that using a small sector size has advantages.

If you plan to store only one or two larger files on the 100K RAM disk, however, you may be better off setting a larger sector size. You realize a time savings when fewer, larger sectors are read or written rather than more, smaller sectors.

The size of the directory also influences the usable size of the RAM disk: the larger the directory, the more room is needed to track files. A 100K RAM disk set up to store 64 files has 98.9K available for storage. The same 100K RAM disk with a directory size of 192 files has only 94.9K for storage. You get a 4 percent larger disk by using the smaller directory size. Practically speaking, you rarely store more than 64 files on a 100K RAM disk, anyway.

A RAM Disk in Expanded or Extended Memory

If your computer has either extended or expanded memory, you may find that a RAM disk is worthwhile. With expanded or extended memory, the actual memory used by the RAM disk is located outside the conventional memory used by programs. You can create a RAM disk without giving up program memory. If you decide to use a RAM disk on a computer with expanded or extended memory, the issue is how large the RAM disk should be and what settings you should use.

If you have expanded memory and use a program like Lotus 1-2-3 Release 2.01 or 2.2, you have more of a decision of how much expanded memory to dedicate to a RAM disk, because 1-2-3 can store data in expanded memory. Creating a RAM disk in expanded memory decreases the amount of expanded memory that 1-2-3 can use. If you have 2M of expanded memory, for example, you may

find you can set aside as much as 512K for a RAM disk without affecting programs that use expanded memory.

If your computer has extended memory, you must decide how much extended memory to use as a RAM disk, if any. Programs such as Lotus 1-2-3 Release 3 use a DOS extender to make extended memory available to the program for data storage. Creating a RAM disk in extended memory, then, decreases the amount of memory available to programs that use DOS extenders.

Regardless of whether you have extended or expanded memory, you also must decide what you can use that memory for in addition to programs. You may want to use some of that memory for SMARTDRV.SYS or IBMCACHE.SYS, for example. If the programs you use are disk oriented rather than RAM oriented (dBASE is disk oriented; 1-2-3 is RAM oriented), you can use expanded or extended memory for a large disk cache. The use of a large disk cache may make a RAM disk obsolete, because the disk cache speeds access to your data files.

Reminder:
You may want to use some extended or expanded memory for disk caches instead of a RAM disk.

If you create a RAM disk in expanded or extended memory, you will probably create a disk larger than one created in conventional memory. When you create a larger RAM disk, you can vary the settings for sector size and directory size. The values you give for these settings, however, still depend on the quantity and size of the files to be stored on the RAM disk.

If you create a 512K RAM disk, for example, you may find that a sector size of 512 bytes is adequate, even if you store small files on the RAM disk. You may also find that a directory size of 64 is adequate because you seldom store more than 64 files on a RAM disk. If, however, you plan to use the RAM disk to help you copy files from one diskette to another, change the directory value to 112. This is the total number of files that can be stored on a 360K diskette.

Cue:
If you use a RAM disk to help copy files between diskettes, set the directory size to 112.

To create a 512K RAM disk with 512-byte sectors and 112 directory entries in expanded memory, include the following directive in CONFIG.SYS:

DEVICE = C:\RAMDRIVE.SYS 512 512 112 /A

To create the same RAM disk in extended memory, include this directive:

DEVICE = C:\RAMDRIVE.SYS 512 512 112 /E

These directives create a RAM disk adequate for temporary storage. Remember the word *temporary*; when the power goes out, you lose what is on the RAM disk. Back up important files on the RAM disk to the hard disk or diskettes.

Controlling Additional Disk Drives with DRIVER.SYS

DRIVER.SYS is a device driver for disk drives. Although DRIVER.SYS is optional on most systems, for some systems that use an external floppy-disk drive, DRIVER.SYS is required.

Cue:
Use DRIVER.SYS to create two logical drives from a single physical drive.

Earlier in this book, you learned that when you have only one floppy-disk drive, DOS establishes two logical disk drives, A: and B:, from the single physical disk drive. You can perform the same kind of "trickery" on any disk drive with DRIVER.SYS.

Cue:
Use DRIVER.SYS to inform DOS that an additional disk drive has been attached to the system.

DRIVER.SYS can be used if you have an external floppy-disk drive or a nonstandard hard drive. You must inform DOS that the additional disk drive is attached to your system. When you invoke DRIVER.SYS for these disk drives, DOS makes them integral parts of the system.

You also can load an additional copy of DRIVER.SYS so that DOS makes two logical disk drives out of any physical disk drive. If drive A: is 1.2M and drive B: is 360K, for example, you can use DRIVER.SYS to make drive A: work as a 360K drive D:.

Before you use DRIVER.SYS, read the information and explanations on the directive's switches, particularly the /D switch. If you do not provide DOS with the needed information, DRIVER.SYS does not work properly.

The syntax for invoking DRIVER.SYS is as follows:

DEVICE = *d:path***DRIVER.SYS /D:ddd** */F:f /T:ttt /S:ss /H:hh /C /N*

The DRIVER.SYS Switches

In the syntax given in the preceding section, *d:* is the disk drive, and *path*\ is the directory path to DRIVER.SYS. Both directives are optional but are usually given. DRIVER.SYS has seven switches, described in the rest of this section.

The **/D:ddd** switch is mandatory and lists the physical disk-drive number. The allowable numbers for **ddd** are 0 to 255. You use the numbers 0 to 127 for floppy-disk drives. The first physical floppy-disk drive, drive A:, is 0; the second, drive B:, is 1; and the third is 2 (the drive name of the third floppy-disk drive varies based on the number of hard drives installed). The values 3 to 127 for floppy disks are possible in theory, but most computers do not support these additional drives. You use the numbers 128 to 255 for hard-disk drives. The first hard disk, drive C:, is 128, and the second is 129. As with floppy disks, the values 130 to 255 can theoretically be used for additional hard-disk drives, but not in fact.

Here is a potential source of confusion in giving disk-drive numbers: If you have an external floppy-disk drive for a PS/2 computer that has a single internal floppy-disk drive, the external drive is the second physical disk drive. Use a 1 for the external disk drive.

If you give a number that corresponds to a nonexistent disk drive, DRIVER.SYS is installed with no indication that anything is wrong. When you attempt to use the disk drive with its logical drive name, however, DOS displays a `General Failure` error. To correct the problem, you must edit the incorrect DRIVER.SYS entry and reboot the computer.

The */F:f* switch identifies the type of disk drive used. /F:f is called the *form factor* switch. The *f* portion of the switch is a single digit taken from the following list:

0	160K to 360K floppy-disk drive
1	1.2M floppy-disk drive
2	720K floppy-disk drive
3	8-inch single-density disk drive
4	8-inch double-density disk drive
5	Hard drive
6	Tape drive
7	1.44M floppy-disk drive

If you do not use the /F:f switch, DOS assumes a form factor of 2.

The */T:ttt* switch indicates the number of tracks per side (or cylinder) for the disk drive. The *ttt* portion of the switch can range from 1 to 999. If you do not specify the switch, a value of 80 tracks is assumed. (1.2M, 720K, and 1.44M floppy-disk drives use 80 tracks; other floppy-disk drives use 40 tracks.) Refer to the documentation that came with your disk drive for the number of tracks used by the drive.

The */S:ss* switch indicates the number of sectors per track. The *ss* portion of the switch can range from 1 to 99. If you do not specify the switch, a value of 9 sectors per track is assumed. (360K and 720K floppy-disk drives use 9 sectors per track.) Refer to the documentation that came with your disk drive for the number of sectors used by the drive.

The */H:hh* switch indicates the maximum number of recording heads per disk drive. The range of *hh* is from 1 to 99. If you do not specify the switch, 2 heads per disk drive are assumed. (Older, single-sided floppy-disk drives use a value of 1; most other floppy-disk drives use 2; hard disks use values higher than 2.)

Never shortchange your disk drive when you give these switches. Use the correct value. For example, don't use /S:8 or /H:1 to read 8-sector, single-sided disks on a 360K disk drive. Don't use /S:9 to read 720K disks on a 1.44M disk drive. Internally, DOS usually knows how to produce a smaller-capacity disk

Cue:
Don't use the DRIVER.SYS switches to set smaller values than your drive is capable of.

from a higher-capacity disk drive. Using settings smaller than your disk drive is capable of simply shortchanges your computer capacity without adding any capabilities.

The */C* switch indicates that the disk drive supports a changeline. A *changeline* lets the disk drive know when the diskette has been ejected or when the floppy-disk drive door has been opened. This switch is used with 1.2M, 720K, and 1.44M disk drives on the PS/2, the AT, and compatibles.

The */N* switch indicates that the disk drive is nonremovable, that is, that the drive is a hard drive. This switch should not be used with floppy-disk drives or with external hard drives.

Table 14.3 lists the DRIVER.SYS switches for common floppy-disk drives.

Table 14.3
DRIVER.SYS Switches for Common Floppy-Disk Drives

Disk Drive	*Switches*
160K/180K	/F:0 /T:40 /S:9 /H:1
320K/360K	/F:0 /T:40 /S:9 /H:2
1.2M	/F:1 /T:80 /S:15 /H:2 /C
720K	/F:2 /T:80 /S:9 /H:2 /C
1.2M	/F:7 /T:80 /S:18 /H:2 /C

Note: The /C switch is meaningful only with the PS/2, AT, and 286-XT. The switch should not be used with other PCs.

Sample DRIVER.SYS Directives

This section presents some examples of using DRIVER.SYS.

If you have a PS/2 computer with two internal disk drives, the syntax to use an external floppy-disk drive is as follows:

```
DEVICE = DRIVER.SYS /D:2 /F:0 /T:40
```

/D:2 designates the third disk drive. This switch is not needed if you have only one internal floppy-disk drive. If you do not specify the /T:40 switch, DOS assumes that the drive uses 80 tracks per side—an error that the 360K disk cannot handle. In this case, the /T switch is required. The /F:0 is also required.

If you want to use the single external physical disk drive as two logical disk drives, load the device driver twice. The syntax for this setup is as follows:

```
DEVICE = DRIVER.SYS /D:2 /F:0 /T:40
DEVICE = DRIVER.SYS /D:2 /F:0 /T:40
```

Each time the driver is loaded, the physical disk drive is assigned an additional valid drive letter automatically—for example, D: the first time and E: the second time. You then may refer to the drive by either of the two new designations.

To use an external 1.2M disk drive on a PS/2 computer, load the driver with this syntax line:

DEVICE = DRIVER.SYS /D:2 /F:1 /S:15 /C

Because the form factor and number of sectors for this disk drive are different from those DOS would establish if you did not specify these switches, you must provide the /F and /S switches. Notice the use of the /C switch, the changeline-support switch. Because PS/2s, ATs, and 286-XTs can recognize and use the changeline of floppy-disk drives, the /C switch is specified.

Changeline support is the reason this directive would be used for an external 720K floppy on an AT or 286-XT. The syntax for this setup is as follows:

DEVICE = DRIVER.SYS /D:2 /C

Use the following DRIVER.SYS syntax for an external 720K floppy-disk drive on a PC or XT:

DEVICE = DRIVER.SYS /D:2

The PC and XT do not recognize disk-drive changelines, so no /C switch is given. Notice that none of the other switches are given because DRIVER.SYS's default settings fit the 720K disk drive.

If the 720K disk drive used in the preceding examples were the second internal drive, you would use 1 for the value of the /D switch. For example, to use an internal 720K floppy-disk drive in an AT, the directive is as follows:

DEVICE = DRIVER.SYS /D:1 /C

Regardless of drive type, /D:1 designates the second internal floppy-disk drive.

You can also load a copy of DRIVER.SYS to use with established disk drives. The copy of DRIVER.SYS, with the appropriate switches, establishes a second logical disk drive for the physical disk.

You do not need this technique if you have a single floppy-disk-drive computer. But you can use the technique when you have two disk drives. Suppose that you have an AT with 1.2M and 360K internal floppy-disk drives. To use drive A: as two logical disk drives, issue the following directive:

DEVICE = DRIVER.SYS /D:0 /S:15 /C /F:1

To use drive B:, the 360K drive, as two logical disk drives, the directive is as follows:

DEVICE = DRIVER.SYS /D:1 /T:40 /F:0

You can use both directives to create four logical drives from the two physical disk drives. If you have one hard disk drive, the "second" 1.2M drive is drive D:, and the "second" 360K drive is drive E:. By setting up your system in this way, you can use either disk drive to copy files between disks.

Naming Disk Drives

Cue: *Understand how DOS names your disk drives.*

The logical disk-drive names that DOS assigns to disks created by DRIVER.SYS and RAMDRIVE.SYS depend on the placement of the directives in the CONFIG.SYS file. You may try to use the wrong disk drive name if you do not know how DOS assigns drive names.

Normally, CONFIG.SYS directives are order-independent. That is, you can give the directives in any order, and the computer does not change how the directive operates.

RAMDRIVE.SYS and DRIVER.SYS do not have the same independence, however. You can load RAMDRIVE.SYS and DRIVER.SYS in any order and the device drivers work, but the disk-drive names change. Whenever DOS encounters a block-device driver (that is, any device that transfers data in blocks, rather than in bytes), DOS assigns the next highest drive letter to the device. The order is first come, first assigned.

If you load a copy of RAMDRIVE.SYS on a system with a single logical hard disk, for example, DOS assigns the name D: to the RAM disk. If you load a second copy of RAMDRIVE.SYS, DOS assigns the name E: to the second RAM disk. Even if the system has only one floppy-disk drive, the first RAM disk is assigned the name E:; the letter B: is reserved for a second floppy-disk drive.

If you load a copy of DRIVER.SYS instead of RAMDRIVE for the external floppy-disk drive on the same computer, DOS assigns the external disk drive the name D:. If you load another copy of DRIVER.SYS for the same external drive, DOS assigns the logical name E: to this disk drive.

Reminder: *Additional partitions of the hard disk are named before drives created by CONFIG.SYS.*

Remember that additional hard disks or additional logical disks add to the starting letter of drives created by CONFIG.SYS. Suppose that the hard drive on a 44M system has the drive names C: and D:. Additional partitions of the hard disk are assigned the next available drive letters before drives created by CONFIG.SYS directives are named. (Hard disks are partitioned with the FDISK command, explained in Chapter 5, and are assigned driver letters whether or not you use DRIVER.SYS.) For this reason, the second partition of the hard disk is named D:. RAMDRIVE, if used, is assigned the drive name E:. DRIVER.SYS, if used instead of RAMDRIVE, is assigned the drive name E:.

The potential for further confusion comes when several block-device drivers are loaded. The order of loading, determined by the order of the directives in the CONFIG.SYS file, determines the names assigned by DOS.

If you load RAMDRIVE.SYS first and DRIVER.SYS second, the RAM disk may be named D:, and the DRIVER.SYS disk is one letter higher (E:). If you switch the lines so that DRIVER.SYS is loaded first, the disk-drive names are switched as well. The DRIVER.SYS disk is named D:, and the RAM disk is named E:.

Reminder:
The order you load RAM disks or DRIVER.SYS disks in CONFIG.SYS determines the names of the disks.

Using Expanded Memory with DOS V4

Two device drivers, XMA2EMS.SYS and XMAEM.SYS, are provided with DOS V4 to enable you to use expanded memory with your system. Although these drivers are intended for IBM memory adapters, you can successfully use the drivers with other brands, including the Orchid RamQuest 50/60 memory add-on.

Note: Some implementations of DOS V4 call the XMAEM.SYS file EMM386.SYS.

Using the XMA2EMS.SYS Driver

You can use XMA2EMS.SYS if you have either an expanded-memory adapter or an 80386-based computer with the XMAEM.SYS driver installed. The syntax used to install this driver is as follows:

DEVICE = XMA2EMS.SYS *FRAME=xxxxP254=yyyyP255=zzzz /X:aa*

xxxx can be any page from address C000 to E000, where the address reflects a 64K page. *P254* and *P255* represent 16K pages of memory used by FASTOPEN and BUFFERS, respectively, to perform their functions in expanded memory (FASTOPEN is described in Chapter 4; BUFFERS is described earlier in this chapter). *yyyy* and *zzzz* are memory addresses of the pages you want to use for these functions.

The */X:aa* switch is used to keep the driver from using less than the maximum total available memory. If you supply a value for *aa* in multiples of 16K pages, XMA2EMS.SYS uses only that amount for expanded-memory specification (EMS). If you use /X:8, for example, 128K of EMS memory is defined for use from the total memory on your expanded- or extended-memory board.

Consult the manuals furnished with your applications programs and peripherals to determine how to allocate expanded memory in your system.

Early users of DOS V4.0 reported serious problems with IBM's EMS support. The V4.0 EMS support implemented expanded memory in a different way than most other available EMS drivers. In particular, you could run into trouble if you told DOS to use expanded memory when you installed FASTOPEN and BUFFERS from CONFIG.SYS. DOS V4.01 corrected this bug.

Using the XMAEM.SYS Driver

The XMAEM driver is designed to be used with 80386-based systems. It enables you to use extended memory that can be addressed as though it were EMS memory. You can use this driver to define the number of 16K pages of extended memory to be devoted to expanded memory.

Note: Some implementations of DOS V4 call the XMAEM.SYS file EMM386.SYS.

If you want to address extended memory as EMS memory, use the following syntax:

DEVICE = XMAEM.SYS *aa*

In this syntax, *aa* is the number of pages to allocate (divide the amount of memory by 16 to calculate this figure). The XMAEM.SYS driver must be loaded before XMA2EMS.SYS in your CONFIG.SYS file. Refer to the software and hardware manuals for your system to install this driver properly.

Displaying and Printing Different Languages

If you want to display more than one code page or print non-English language characters, you must use the DISPLAY.SYS and PRINTER.SYS device drivers. (Code pages are described fully in Chapter 15.)

Reminder: *Use DISPLAY.SYS and PRINTER.SYS to display and print foreign-language characters.*

Using DISPLAY.SYS in your CONFIG.SYS file enables you to switch code pages without restarting DOS. Using PRINTER.SYS enables you to download a font table to certain printers so that they can print non-English language and graphic characters.

DOS V3.3 and V4.0's support of different languages depends not only on these two device drivers, but also on the NLSFUNC program and on the CHCP, MODE CODEPAGE PREPARE, MODE CODEPAGE SELECT, MODE CODEPAGE REFRESH, and KEYB commands. All of these commands are discussed in Chapter 15.

Displaying International Characters with DISPLAY.SYS

The DISPLAY.SYS driver enables the switching of code pages on most video adapters. DISPLAY.SYS is ineffective with a monochrome or color-graphics adapter: the code pages of these adapters are fixed and cannot be switched. If you have a monochrome or color-graphics adapter, DISPLAY.SYS is not recommended for your use. With DOS V4, DISPLAY.SYS has been enhanced to check your hardware to determine the type of display you have if you don't specify the adapter type.

You can use DISPLAY.SYS effectively with these adapters: PC Convertible LCD, EGA, and VGA.

The syntax of the line you add to the CONFIG.SYS file to invoke the DISPLAY.SYS driver is as follows:

DEVICE = *d:path***DISPLAY.SYS CON**:* **= (type**, *hw_codepage, added_codepages, subfonts*)

Code pages are discussed more completely in Chapter 15 and are beyond the scope of this brief device-driver overview. The following paragraphs, however, review the parameters shown in the directive.

The *d:path*\ parameter is the drive name and path to the DISPLAY.SYS file. If DISPLAY.SYS is not in the root directory of your starting disk, supply the drive and path name.

CON: is the DOS device name for the console—the keyboard and display. The colon after the device name is optional.

Following **CON** is an equal sign and, in parentheses, the parameters to be passed to DISPLAY.SYS. The equal sign and enclosing parentheses must be specified. The *type* parameter can be any of the following:

Parameter	*Meaning*
MONO	Monochrome adapter
CGA	Color-graphics adapter
EGA	Enhanced color-graphics adapter or PS/2
LCD	Convertible LCD

The second parameter passed to DISPLAY.SYS is *hw_codepage*, which stands for the hardware code-page built into your adapter (437, 850, 860, 863, or 865). DISPLAY.SYS uses the code page given to the COUNTRY directive. (COUNTRY is discussed in Chapter 15.)

The *added_codepages* parameter is the maximum number of additional code pages that the adapter can use. This parameter actually refers to the number of code pages that can be prepared and used with the MODE command. Refer to Chapter 15 for a full description of changing the code pages with MODE.

The *subfonts* parameter refers to the dot resolution of the character displayed. Characters can be formed from a box of 9 by 14 dots, 9 by 16 dots, or 8 by 8 dots. The subfonts value can range from 0 to 2, depending on the adapter you use. If you use a monochrome or CGA adapter, set subfonts to 0. (0 is the value DOS assumes if you do not specify a subfonts value.) If you use a convertible LCD display, set subfonts to 1. (1 is the value DOS assumes if you don't specify subfonts for the LCD display.) If you use an EGA or VGA adapter, set subfonts to 1 or 2. (DOS defaults to 2 if you use an EGA adapter and don't specify this parameter.)

If you use an EGA or VGA monitor and want to specify 437 as the hardware code page, include this command in CONFIG.SYS:

```
DEVICE = C:\SYS\DISPLAY.SYS CON = (EGA,437,1)
```

This command specifies an EGA display (you specify EGA if you have an EGA or a VGA display), 437 for the starting code page, and 1 additional code page. You don't have to use additional code pages, but if you plan to, specify a value for *added_codepages.*

The following command specifies an EGA display and a code page of 863 (French-Canadian):

DEVICE = C:\SYS\DISPLAY.SYS CON = (EGA,850,863,2)

Notice that the existing code page is 863, but that the specified code page is 850. If your existing code page is not 437, always use 850 as the new code page and 2 as the value for added code pages.

Reminder: *Include the ANSI.SYS directive in CONFIG.SYS before the DISPLAY.SYS directive.*

If you use ANSI.SYS and DISPLAY.SYS in your CONFIG.SYS file, the ANSI.SYS directive must appear before the DISPLAY.SYS directive. If the order is reversed, you lose the capability of ANSI.SYS. Remember that you do not have to include DISPLAY.SYS if you do not use code-page switching. If you are new to DOS and do not use a font different from the one built into your display, you do not have to include the DISPLAY.SYS driver in your CONFIG.SYS file.

Printing International Characters with PRINTER.SYS

PRINTER.SYS is the device driver that enables the characters shown in the code page to be printed on certain printers. DOS V3.3 supports code pages on only two printers:

IBM ProPrinter Model 4201
IBM Quietwriter III Model 5202

DOS V4 adds support for these printers:

IBM ProPrinter Model 4202
IBM ProPrinter Model 4207
IBM ProPrinter Model 4208

These printers are the only printers on which DOS can control the printing of foreign-language characters; other printers are controlled in other manners. To use code-page switching with these printers, you must add PRINTER.SYS to your CONFIG.SYS file. The syntax for PRINTER.SYS is similar to that for DISPLAY.SYS:

DEVICE = *d:path***PRINTER.SYS LPT***x:** = (**type**, *hw_codepage, added_codepages*) . . .

or

DEVICE = *d:path***PRINTER.SYS LPT***x:** = (**type**, *(hw_codepage1,hw_codepage2), added_codepages*) . . .

Notice that on one line, you must specify code pages for printers on multiple parallel ports (designated by the three-dot ellipsis). Also note that you do not type the parentheses around *hw_codepage1,hw_codepage2*; the parentheses are shown in this syntax line to ease readability.

This section explains each element in these lines and then gives sample lines to show how the directive varies for each printer.

The *d:path*\ parameter is the drive name and path to the PRINTER.SYS file. If PRINTER.SYS is not in the root directory of your starting disk, give the disk drive and path name.

The **LPTx:** parameter is the DOS device name for the line printer. The **x** is the number of the parallel printer port: 1, 2, or 3 (DOS does not support serial printers when printing international characters). The colon after the device name is optional. Because you can have up to three parallel printers, you must specify the parallel port to which the printer is connected by including up to three DEVICE=PRINTER.SYS lines in CONFIG.SYS.

Reminder:
You can specify up to three parallel printers.

Following the **LPTx**: parameter are an equal sign and the parameters passed to PRINTER.SYS, enclosed in parentheses. The equal sign and enclosing parentheses must be given. The printer **type** is the model number for the printer: 4201 for the ProPrinter (with DOS V3.3) or 4202, 4207, or 4208 (with DOS V4), or 5202 for the Quietwriter III (DOS V3.3 or V4).

The *hw_codepage* parameter stands for the hardware code-page built into your printer. As with DISPLAY.SYS, the possible values for this parameter are 437, 850, 860, 863, and 865. Following is a chart explaining these values.

Code Page	*Meaning*
437	United States English
850	Multilingual
860	Portuguese
863	French-Canadian
865	Norwegian

The *added_codepages* parameter is the maximum number of additional code pages that the printer can use. The value for this parameter can range from 0 to 12.

Following is an example of the directive to use with a ProPrinter:

```
DEVICE = C:\SYS\PRINTER.SYS LPT1 = (4201,437,1)
```

This directive specifies a ProPrinter 4201 connected to the first parallel port (LPT1), 437 as the starting code page, and 1 as the additional code page. You don't have to use additional code pages, but if you plan to, specify a value for *added_codepages*.

This example shows that the PRINTER.SYS file is in a subdirectory on drive C: called \SYS. If PRINTER.SYS is in a different directory, change the disk drive and path information but leave all other information the same.

Remember that you do not have to include PRINTER.SYS in CONFIG.SYS if you do not use code-page switching with the DOS-supported printers. If you do not use a font different from the one built into your printer, or if you do not use one of the supported printers, you can ignore the PRINTER.SYS device driver.

DOS V3.3 and V4 users receive two additional SYS files: COUNTRY.SYS and KEYBOARD.SYS. These files are not device drivers. Do not attempt to use them with the DEVICE directive. These files are discussed in Chapter 15.

Accessing Files through FCBS

Cue:
FCBS enables you to use programs written for DOS V1 with new versions of DOS.

Some DOS users find the FCBS directive indispensable. It enables you to use useful, but antiquated, programs written for DOS V1 with later versions of DOS.

FCB is an acronym for *file control block*. FCBs are one way a program can access a file. This method of file access was used by DOS V1 to communicate with programs. Later versions of DOS borrow a UNIX-like method for controlling files, called *handles* (discussed in "Using the FILES Directive" later in this chapter). Although FCBs can be used with any version of DOS, handles can be used only with DOS V2, V3, or V4.

To use the FCBS directive in the CONFIG.SYS file, use this syntax:

FCBS = maxopen, *neverclose*

In this syntax, **maxopen** is the maximum number of unique FCBs that programs can open at one time, and **neverclose** is the number of FCBs that DOS cannot reuse automatically. You must specify maxopen, which can be a number from 1 to 255. The default value is 4. The optional *neverclose* parameter must be a number less than or equal to **maxopen**. If you don't specify *neverclose*, the default value is 0. Set these values by trial and error; if you set values, run the program, and find that the program cannot open all the required files (a message to this effect appears), set the FCBS parameters to higher values.

Although the IBM documentation states that FCBs are closed only when you use the SHARE command for a local area network, this is not true with DOS V3.0 and V3.1. If you use an older program that doesn't support path names, the program uses FCBs, and you should set the FCBS directive.

You pay a small price in RAM to use the FCBS directive. For each number above 4 that **maxopen** exceeds, DOS uses about 40 bytes. Considering that most people have 256K or more of RAM in their computers, the use of this extra RAM should not be a problem.

Using the FILES Directive

As the FCBS directive decreases in importance with the later versions of DOS, the FILES directive increases. FILES is the DOS V2, V3, and V4 directive used to enable UNIX-like or XENIX-like file handling. UNIX and XENIX—and later versions of DOS—use a file *handle* (a number corresponding to the file name) instead of file-control blocks to access files. Your program gives the operating system the name of the file or device you want to use. The operating system gives back to your program a handle—a two-byte number. From that point on, your programs use the handle, rather than the FCB, to manipulate the file or device.

To include the FILES directive in CONFIG.SYS, use the following syntax:

FILES = nn

In this syntax, **nn** is the number of files you want opened at any time. The maximum is 255, and the minimum is 8. If you give a FILES directive with a number less than 8, or if you omit this directive, DOS makes the number 8. Each additional file over 8 increases the size of DOS by 39 bytes.

Cue:
Set FILES to at least 10.

The name FILES is somewhat deceiving, because handles are used by both files and devices. For example, standard devices such as the display, keyboard, or printer automatically receive one file handle each. If you do not specify the FILES directive, DOS starts with eight file handles and immediately takes five handles for the standard devices, leaving only three handles for your programs. Including this directive in your CONFIG.SYS file is a good idea:

FILES = 10

This directive establishes 10 file handles, which should be enough for most programs. If a program displays an error message about file handles, edit CONFIG.SYS and increase the number of handles to 15 or 20 (if you are using dBASE III Plus, use 20 files). These higher numbers are sufficient for most existing programs.

Using LASTDRIVE To Change the Number of Disk Drives

The LASTDRIVE directive informs DOS of the maximum number of disk drives on your system. Generally, LASTDRIVE is a directive used with networked computers or with the pretender commands (such as SUBST).

If you don't use the LASTDRIVE directive, DOS assumes that the last disk drive on your system is E:. If you give DOS a letter corresponding to fewer drives than are physically attached to your computer, DOS ignores the directive. The

LASTDRIVE directive enables you to tell DOS how many disk drives, real or apparent, are on your system. Suppose that you have a system with two floppy-disk drives and a hard disk partitioned into three drives (C:, D:, and E:). If you issue the directive **LASTDRIVE=D**, DOS ignores the directive because it knows that the last drive is E:.

If you want to use the LASTDRIVE directive in CONFIG.SYS, use the following syntax:

LASTDRIVE = x

In this syntax, **x** is the alphabetical character for the last disk drive on your system. The letters A through Z, in uppercase or lowercase, are acceptable.

Cue:
Use LASTDRIVE to let DOS know you have created a RAM disk, a virtual disk, or a logical disk.

Why would you want to use the LASTDRIVE directive? One reason is so that you can use multiple RAM disks. Suppose that you have a system with two floppy-disk drives (drives A: and B:) and a hard drive (drive C:). You create two RAM drives (drives D: and E:). To create a third RAM drive, you must issue the directive **LASTDRIVE=F** so that you can control the third RAM drive as drive F:.

Another reason you may want to use LASTDRIVE is to establish logical disk drives. A *logical* disk drive can be a nickname for another disk drive (see the SUBST command). A logical disk drive may also be another partition of the hard disk. A logical disk drive is just a name. DOS "thinks" that the logical disk drive is real.

Using the SHELL Directive

The SHELL directive was originally implemented to enable programmers to replace the DOS command processor (COMMAND.COM) with other command processors. The directive now has two additional functions: You can place the command processor in any directory—not just in the boot disk's root directory—and you can expand the size of the environment.

SHELL is a tricky directive that should be used cautiously. Giving the wrong SHELL directive can "lock up" your system. Until you understand the directive, don't use it in your CONFIG.SYS file.

Cue:
When you change the SHELL directive, have handy another DOS diskette to use if you make mistakes.

When you do need to use the directive to move COMMAND.COM from the root directory or to increase the size of the environment, have handy another DOS diskette that can start up your computer. If you make a mistake, reboot with this diskette.

You may want to keep COMMAND.COM in a directory other than the root directory to make it harder for computer "viruses" to locate the file. Another reason is to protect the file when you copy a diskette containing another copy

of COMMAND.COM. If you accidentally copy the diskette to the root directory, you won't overwrite your version of COMMAND.COM.

The SHELL directive is slightly different for DOS V3.0 and later versions. The syntax for the SHELL directive in DOS V3.0 is as follows:

SHELL = *d:path***filename**.*ext d:\path /P*

The syntax for SHELL for DOS V3.1 through V4.0 is as follows:

SHELL = *d:path***filename**.*ext d:\path /P /E:size*

In these syntaxes, *d:path*\ is the name of the disk drive and the subdirectory path to the command processor you want to use. *d:path*\ is optional but should be given. **filename.ext** is the name of the command processor. The root name is required, and the extension is needed only if the file does not use the extension .COM.

The second *d:\path* parameter is the disk drive and path name to the command processor. DOS uses this information to set the COMSPEC environment variable. This drive and path name should duplicate the first disk drive and path name. Although this drive and path are optional, they should be given.

DOS versions before V4.0 give SHELL two switches:

/P

/E:size

DOS V4 adds a new switch to COMMAND.COM that can be used with the SHELL:

/MSG

The */MSG* switch tells DOS to load DOS messages into RAM when it loads COMMAND.COM. This enhancement can speed up operation somewhat and increases the amount of memory DOS uses only by about 1K.

The */P* switch instructs DOS to load and keep resident a copy of the command processor. Without the /P switch, DOS loads COMMAND.COM, executes the AUTOEXEC.BAT file, and immediately exits. You then must reboot the system using a different disk if you omit the /P switch.

/E:size is an optional switch that sets the amount of random-access memory used for the environment. If you see the `Out of environment space` message, use the SHELL directive with the /E switch to specify a larger space for environment variables.

If you use /E, the size of DOS increases by the amount of space you add to the normal starting value for the environment. The SHELL directive does not use any additional memory except when it increases the size of the environment.

Reminder:
SHELL does not increase the size of DOS unless you specifically make the environment larger.

DOS V3.2, V3.3, and V4.0 start with an environment size of 160 bytes. You can specify a size from 160 to 32,767. If you give a size number greater than 32,767, DOS uses 32,767. If you give a number smaller than 160 or give a nonsense size (such as a letter), DOS uses 160 bytes for the environment.

If the size you specify is not evenly divisible by 16, DOS adjusts it to the next higher multiple of 16. For example, if you specify 322, DOS automatically adjusts the size to 336.

DOS V3.1 uses a different system for size: the number of 16-byte paragraphs for the environment. DOS V3.1 starts with an environment size of 128 bytes. You can change the size with the SHELL directive from 11 (11 × 16, or 176 bytes) to 62 (62 × 16, or 992 bytes). If you use a number less than 11 or greater than 62, DOS V3.1 displays an error message and ignores the /E switch.

Using the STACKS Directive

The STACKS directive was added to DOS V3.2 and was changed slightly with V3.3. If you do not have an Enhanced Keyboard, this directive may have little use for you. If you have a PS/2, AT, or 286-XT, you may find this directive useful. To understand why, you need to know a few things about your computer's internal operation.

Reminder: *The CPU uses stacks to store information it needs to return to when it is interrupted.*

A *stack* is an area of RAM. The CPU (central processing unit, or "brain" of your computer) uses this area to temporarily hold information. High-speed instructions place items onto the stack and remove items from the stack. The IBM's CPU can use a stack anywhere in memory, and each program you use can have its own stack.

The CPU often receives *hardware interrupts* (messages sent by the keyboard, mouse, disk drive, or the like that demand the computer's attention). When a hardware interrupt causes the computer to stop what it was doing to respond to the interrupt, the CPU stores the information being acted on when the interrupt occurred. The CPU stores this information in a stack.

Unfortunately, it's possible to interrupt an interrupt, so the stack can be filled quickly. Versions of DOS before V3.2 used a single-stack method to handle interrupts. When too many interrupts occurred and the stack overflowed, the system would mysteriously lock up, and no error message would be given.

DOS V3.2 and later versions use a pool of nine stacks. When a hardware interrupt occurs, DOS draws another stack from the pool. When DOS has successfully answered the interrupt, it returns the stack to the pool for use later when another interrupt occurs. Overall, the lock-up problems have been reduced considerably.

Today, most stack problems surface when you use an Enhanced Keyboard and certain programs, such as dBASE III Plus. Holding down a key can trigger stack failure. The electronics in an Enhanced Keyboard send signals more quickly than does a standard keyboard, making the stack fill up too quickly.

To prevent the problem of stack overflows, DOS provides the STACKS directive. The syntax for this directive is as follows:

STACKS = number, size

In this syntax, **number** is the number of stacks that DOS is to establish to handle the interrupts. The value of **number** should be from 8 to 64. The **size** parameter is the size in bytes of each stack. This value can be from 32 to 512.

Cue:
Use STACKS to specify how many stacks DOS is to use and how big the stacks are to be.

The default value of the STACKS directive varies by machine. For DOS V3.2, V3.3, and V4.0 on a PS/2, AT, XT, and 286-XT (with an Enhanced Keyboard), as well as the PC Convertible, DOS starts with 9 stacks of 128 bytes each (equivalent to giving the directive STACKS=9, 128). DOS V3.3 on the PC, XT, and PC Portable has a single stack for all interrupts.

To revert to the "single-stack-fits-all" approach for handling interrupts, use this directive:

STACKS = 0, 0

Generally, you don't have to worry about the STACKS directive. If you get a stack error, however, you are using a program that triggers the hardware problem. Immediately change the CONFIG.SYS file to increase the number of stacks without altering the size of the stacks. A comfortable line to use is one that establishes 12 stacks of 128 bytes each:

Reminder:
If you get a stack error, increase the ***number*** *parameter in the STACKS directive.*

STACKS = 12, 128

The preferred way to handle stack failure is to increase the number of stacks and not to increase their size. If, however, you must change the size of the stacks, do so with care. Do not decrease the size of the stacks; any benefits you receive by decreasing the size of the stacks are greatly outweighed by the chance that a program will overflow the smaller stack.

Understanding DOS V4 CONFIG.SYS Commands

In addition to the new device drivers, DOS V4 provides three additional CONFIG.SYS directives: REM, SWITCHES, and INSTALL.

Using the REM Directive

Cue:
Use REM to insert remarks or keep the processor from executing other directives.

REM enables you to insert remarks in your CONFIG.SYS file. You can leave notes to yourself (or others) explaining what particular lines do. This kind of documentation in a CONFIG.SYS file is especially helpful if you use non-DOS device drivers for your hardware. You also can remove temporarily a CONFIG.SYS statement by prefacing it with a REM directive. When you test a new configuration, you can enable only those features you want by removing the word *REM*.

To use the REM directive, use the following syntax:

REM *remarks*

Using the SWITCHES Directive

SWITCHES turns off Extended Keyboard functions. This directive works like the ANSI.SYS /K switch (described in Chapter 12). The syntax for this directive is as follows:

SWITCHES = /K

Some software cannot work with the Extended Keyboard. If you use this directive to disable the Extended Keyboard for that software, the software should function properly.

Using the INSTALL Directive

INSTALL enables you to load from the CONFIG.SYS file certain utility programs that remain in memory. In versions of DOS before V4, you had to load these programs from the DOS prompt or through a batch file such as AUTOEXEC.BAT. DOS V4 supports loading any of the following programs through the use of INSTALL:

FASTOPEN.EXE
KEYB.COM
NLSFUNC.EXE
SHARE.EXE

The Command Reference can provide more information about these programs. Loading these programs from CONFIG.SYS instead of from AUTOEXEC.BAT can result in a memory savings of several kilobytes per program.

The syntax for using INSTALL in CONFIG.SYS is as follows:

INSTALL = *d:path***program** *parameters*

In this syntax, *d:path*\ is the disk and path information, and **program** is the name of the utility you want to load. You may be able to use INSTALL with some non-DOS utilities such as SideKick and Keyworks. The program you install with this directive must have the extension .COM or .EXE; you cannot install a file such as a batch file.

Gaining More Speed with FASTOPEN

Cue:
Use FASTOPEN if BUFFERS doesn't make your hard drive act fast enough.

Although FASTOPEN is not a CONFIG.SYS directive, it seems appropriate to discuss it here after the INSTALL directive. FASTOPEN, introduced with DOS V3.3, can be used only with hard drives. Whereas buffers store information moved from disk to user memory, FASTOPEN allocates memory to store directory information. FASTOPEN partially solves a performance problem not resolved by the CONFIG.SYS BUFFERS directive. BUFFERS helps when your computer reads or writes information to several files, and helps most when disk activity is isolated to a file's few key portions.

If you use many of the same files during the day, however—particularly small files that DOS can read or write in one cluster—BUFFERS is not a major benefit. The time DOS spends traversing the subdirectory system and opening the files may take more time than actually reading or writing the files.

FASTOPEN caches directory information, holding in memory the locations of frequently used files and directories. Directories are a type of file, not accessible by users, that DOS reads and writes in a manner similar to that of other files. A part of the directory entry for a file or subdirectory holds the starting point for the file in the file allocation table (FAT). Because DOS typically holds the FAT in the disk buffers, FASTOPEN was developed to hold directory entries in memory.

FASTOPEN is not a complex command, but you must do a little work before you can use it effectively. FASTOPEN's syntax is as follows:

*INSTALL d:path***FASTOPEN.EXE d: = nnn**

or

*d:path***FASTOPEN**.*EXE* **d: = nnn**

You use *INSTALL* and the *.EXE* extension only if you use this line in CONFIG.SYS. If you issue this command at the DOS prompt, omit these items and type the rest of the command.

The *d:path*\ parameter is the disk drive and path to the FASTOPEN.EXE file. The **d:** following the file name is the name of the hard drive you want FASTOPEN to act on and is a mandatory part of the command. The **nnn** parameter is the number of directory entries that FASTOPEN should cache. Each file

or subdirectory requires one directory entry, and you can enter a value from 10 to 999. If you do not specify a value for **nnn**, DOS defaults to 34.

If you use DOS V4, you can use several additional parameters with this command, as shown in the following syntax line:

*INSTALL d:path***FASTOPEN.EXE d: = (nnn**,*mmm*) */X*

or

*d:path***FASTOPEN**.*EXE* **d: = (nnn**,*mmm*) */X*

The new *mmm* parameter refers to the number of fragmented entries (the DOS manual calls them "continuous space buffers") for the drive. Using this parameter improves performance when files on your hard disk become fragmented. The values you can use for *mmm* range from 1 to 999. DOS has no default value for fragmented entries. If a value is not given, the feature is not active.

The */X* switch is similar to the /X switches of other commands. This switch enables FASTOPEN information to reside in expanded memory.

Cue:
Use FASTOPEN to speed performance on as many hard disks as you have on your computer.

You can use FASTOPEN on as many disks as you want. Note, however, that the total number of directory entries or fragment entries FASTOPEN can handle is 999. If you issue the command for several disk drives, the sum of **nnn** values or the sum of *mmm* values cannot exceed 999. This limit is FASTOPEN's physical limit.

The practical limit of **nnn** is between 100 and 200 per disk. If you specify a value much higher, DOS wades through the internal directory entries slower than it reads information from the disk. Additionally, each directory entry stored in memory takes 35 bytes. Considering the speed-and-memory trade-off, the 100-to-200 limit yields adequate performance.

Using too small a number for **nnn** can be a disadvantage as well. When directory entries are recycled, the least recently used entry is discarded if a new entry is needed. If the **nnn** value is too small, DOS discards entries it may still need. The object is to have enough entries in memory so that FASTOPEN operates efficiently, but not so many that FASTOPEN wastes time wading through directory entries.

At the very least, **nnn** must exceed the number of subdirectories through which you travel to get to the "deepest" subdirectory. The minimum value for **nnn** is 10; this value often exceeds the number of levels in your directory organization. Suppose that you have a directory structure like \DOS\BASIC\TEST. The deepest "level" is 3 down from the root, much less than DOS's default of 10. A not terribly scientific guideline is to start with 100 and fine-tune up or down by 5 until you "feel" the right performance.

The value you specify for *mmm* depends on how fragmented your files are. Use the CHKDSK *.* command in subdirectories where you store data files to get a

report of fragmentation. The more fragmentation, the larger the value you should assign to *mmm*. A general rule is to use a value for *mmm* that is twice the number of **nnns** as your *mmm* entry.

Reminder: *Install FASTOPEN after you define all disk drives.*

You should observe two restrictions about using FASTOPEN. First, the disk drive you name cannot be one on which you use JOIN, SUBST, or ASSIGN, because these commands do not create "real" drives. Second, if you load a disk-drive device driver through AUTOEXEC.BAT rather than through CONFIG.SYS (some manufacturers provide a driver that must be started from a batch file or from the DOS prompt rather than from CONFIG.SYS), you must use FASTOPEN *after* you have defined all disk drives. FASTOPEN can become confused if you add additional disk drives after you have invoked FASTOPEN.

Making a CONFIG.SYS File

Following is a sample CONFIG.SYS file for an IBM AT or PS/2 Model 60:

```
DEVICE = C:\SYS\SJDRIVER.SYS /M
DEVICE = C:\SYS\HPSCANER.SYS
DEVICE = C:\SYS\XMA2EMS.SYSFRAME=C000 P254=D000 P255=D400 /X:32
DEVICE = C:\SYS\IBMCACHE.SYS 1900 /E /P8
LASTDRIVE = G
BUFFERS = 10,5 /X
COUNTRY = 001,,C:\SYS\COUNTRY.SYS
DEVICE = C:\SYS\DISPLAY.SYSCON=(,437,1)
SHELL = C:\DOS\COMMAND.COM C:DOS /P /MSG
DEVICE = C:\SYS\ANSI.SYS /L
REM DEVICE = C:\SYS\RAMDRIVE.SYS 360 /X:8
INSTALL C:\DOS\FASTOPEN.EXE C:=(100,50) /X
FILES = 30
```

Examining this specialized file can provide you with some useful tips for building a customized CONFIG.SYS file of your own. This sample file assumes that you have a subdirectory called \SYS on your hard disk and that device drivers are in this subdirectory.

The first two lines load special third-party device drivers needed to operate an HP ScanJet scanner. Consult the syntax recommendations of your software or hardware vendors to see exactly how to phrase your own DEVICE directives when installing such drivers.

The LASTDRIVE and BUFFERS commands were explained earlier in this chapter. Only 10 buffers are used because the IBMCACHE program reduces the need for a larger number of buffers.

Several other directives are in the CONFIG.SYS file for various reasons. For example, the COUNTRY directive and DEVICE=DISPLAY.SYS statement are

given to use code pages. The SHELL directive tells DOS where the command processor is located, passes the directory of the command processor to COMSPEC, and makes memory-resident the messages COMMAND.COM uses. ANSI.SYS is loaded as a device driver so that ANSI-compatible programs can use ANSI.SYS to control the screen. FILES is set to 30 to allow up to 30 files to be open at once.

Reminder: *Use REM to "comment out" directives you don't need all the time.*

Notice the line where RAMDRIVE.SYS is loaded. In the listing shown, the line is commented out using the REM directive. When you use a directive only on occasion, you may find it convenient to leave the directive in the CONFIG.SYS file, preceding it with the REM directive so that DOS "ignores" the rest of the line. When you need a RAM disk, simply edit the CONFIG.SYS file and remove REM from the line. When you do not want to use the directive, edit the CONFIG.SYS file and add REM to the beginning of the directive. The alternative is to type the complete directive whenever you need it and to erase it completely when you don't need it.

Summary

In this chapter, you learned the following important points:

- ❏ DOS can alter system settings through instructions in the CONFIG.SYS file.
- ❏ The CONFIG.SYS file must be in the root directory of the boot disk. When you alter CONFIG.SYS, the changes do not occur until DOS is rebooted.
- ❏ You can tell DOS when to look for the Ctrl-Break key sequence.
- ❏ Disk buffers make DOS work faster by placing requested information in RAM.
- ❏ Disk caches like the DOS IBMCACHE program can speed up certain disk operations.
- ❏ RAMDRIVE, the DOS RAM-disk software, can speed disk operations if your computer has sufficient random-access memory.
- ❏ DRIVER.SYS, the device driver for disk drives, must be used for external or non-IBM hard disk drives. DRIVER can also be used to make additional logical disk drives from a physical disk drive.
- ❏ DOS V4 enables the use of expanded memory.
- ❏ DISPLAY.SYS and PRINTER.SYS are used to display and print foreign-language characters.

- DOS V1 used a different method of accessing files; this method can be reproduced with later versions of DOS through the FCBS directive.
- The FILES directive sets the number of files DOS can open at any one time.
- The LASTDRIVE directive specifies the letter of the last disk drive installed in your system.
- You can replace the command processor DOS uses by invoking the SHELL directive.
- You can specify the number of stacks available to the processor during an interrupt with the STACKS directive.

In Chapter 15, you learn about the files and directives that enable you to use the international features of DOS.

15

Understanding the International Features of DOS

The Personal Computer, PS/2 computers, and DOS are American inventions. But DOS is an international operating system. Vast changes in DOS V2.11 made the program easier to use in countries outside the United States. Additional changes occurred in each revision of DOS V3, the most radical changes since DOS V2.11 being in DOS V3.3. DOS V4 added some minor enhancements. If you use your computer in a country other than the United States, these changes make your computer more natural to operate.

Cue:
With DOS V3 and V4, the date and time can be displayed in one of three formats.

DOS V3 and V4 have several international features seldom noticed by users in the United States. One feature is that the date and time can be displayed in three different formats. In addition, DOS can tell a program, on request, which characters to use for the decimal point, the thousand separator, and the currency symbol. DOS can even tell the program whether the currency symbol precedes or follows the amount and whether a space should separate the symbol from the amount.

Furthermore, the displays and keyboards of Personal Computers and PS/2s can be changed to produce characters that are appropriate for countries other than the United States. Although users outside the United States have had these programs since DOS V1.1, IBM now distributes the programs as part of DOS V3 and V4 in the United States.

To make DOS go international, you must alter your CONFIG.SYS and AUTOEXEC.BAT files. These changes are recommended only for those who need international capability. If you do not need such capability, you can skip this chapter until you do.

To customize DOS for international use, you may need to perform several actions:

1. Add the following items to your CONFIG.SYS file:
   a. The COUNTRY directive
   b. The DEVICE = DISPLAY.SYS directive
   c. The DEVICE = PRINTER.SYS directive
2. Add the NLSFUNC, MODE, CHCP, and/or KEYB commands to your AUTOEXEC.BAT file.

To understand how international-character support works in DOS, you first need to understand the concept of *code pages*.

Understanding Code Pages

Code pages enable the computer to use, display, and print non-English language characters with minimal effort on your part.

You may recall from earlier discussions that your computer uses ASCII to communicate to peripherals. ASCII is the standard manner of translating the numeric values 0 to 255 into letters, numbers, and symbols. On your screen the dots that form the visible characters are the graphic representations of those ASCII values.

Think of your computer as having three punchboards, one each for the keyboard, video display, and printer. Each punchboard has holes arranged in a 16-by-16 square, for a total of 256 holes. You can move to any hole, punch it, and get the message ("prize") inside. For example, if you punch the hole marked 65, out pops the character whose ASCII value is 65—the letter A.

When you press the A key, your computer punches the video adapter's hole for 65. The adapter, in turn, looks up the numeric value and finds the pattern of dots that corresponds to the character. The adapter turns on the appropriate dots, and the letter A appears on-screen.

Cue:
A font is a particular "shape" of characters.

Each video adapter displays a dot-formed representation of ASCII characters. The common name for the "shape" of characters is *font*. Depending on the adapter in use, the fonts are switchable, meaning that different character sets can be displayed.

Cue:
A code page is a font set.

Basically, a code page is a font set. You may recognize names for typewriter and typesetting faces, such as Courier, Elite, and Times Roman. Similarly, code pages allow for different typefaces of English and international language characters.

Five video adapters are available: the Monochrome Display Adapter (MDA), the Color Graphics Adapter (CGA), the Enhanced Graphics Adapter (EGA), the Liquid Crystal Display (LCD) of the PC Convertible laptop, and the Video Graphics Array (VGA) of the PS/2. (LCDs and VGAs are not necessarily adapters; some LCDs and VGAs are integrated circuitry built into the computers and are responsible for the display of information.)

The VGA, EGA, and LCD adapters can use the five international font sets listed in table 15.1. Depending on the code page ("punchboard") used, you can display various language characters, as indicated in the table. Monochrome and CGA adapters cannot support switchable code pages.

Table 15.1
Code Page Numbers

Font	*Code Page Number*
United States (English)	437
Multilingual	850
Portuguese	860
Canada (French)	863
Nordic	865

Suppose, for example, that you punch the hole for ASCII character 236. Each font set except the Multilingual code page displays the infinity character. The Multilingual code page displays an accented Y. Punch position number 157 on the North American set, and you get a double-bar Y. Punch the same position on the Multilingual or Norwegian-Danish set, and you get a 0 (zero) with a slash through the number. The Portuguese and French Canadian sets have a back-accented U in this position. Many characters are identical in all five sets, but some are customized for various localities.

Appendix F contains the five code pages used by PCs and PS/2 computers. The major difference among these code pages is the characters whose values are in the range 128 through 255 (columns 9 through F on the charts). No widely used, prevailing standard exists for these characters. DOS sacrifices some of the scientific and graphics characters in this range when the program substitutes the needed language characters.

Printers also use code pages; what can be displayed on-screen can therefore be reproduced on the printed page. DOS V3.3 includes code pages for IBM ProPrinter Model 4201 and Quietwriter III 5202. DOS V4 supports those printers as well as IBM ProPrinter Models 4202, 4207, and 4208. You can expect other printer manufacturers to follow the lead of Microsoft and IBM in building code page support into DOS and to provide code pages for their printers in the near future. Many printers now emulate to some degree one of these IBM printers. Check your printer manual to determine the printer's code page compatibility.

Reminder:
Because printers use code pages, what you can display on-screen can be reproduced on paper.

Reminder:
Select a starting code page when you want to use a particular language font.

To use a particular language font, you select a starting code page; you can change code pages freely. You may find, though, that code page switching is completely unnecessary if your video adapter and printer start with the "right" code page. Most North American users find the built-in DOS code pages adequate and probably never use code page switching.

Switching code pages may be important for international users, however, including those who have files established in early versions of DOS. In addition, the ability to change fonts "on the fly" can be valuable. International users will find that code pages provide a logical and powerful way to handle languages.

Using the International Directives of DOS

DOS uses directives and device drivers to set up your computer so that it uses a particular character set. The directives are COUNTRY, NLSFUNC, MODE (in three forms), and CHCP. The device drivers are KEYB.SYS, PRINTER.SYS, and DRIVER.SYS. Some directives and device drivers are included in the CONFIG.SYS file. Other commands required for international support can be included in your AUTOEXEC.BAT file, enabling them to run each time you boot your system.

Changing the CONFIG.SYS File

You include in your CONFIG.SYS file the COUNTRY directive and two drivers that control devices attached to your system: DISPLAY.SYS for the display and PRINTER.SYS for the printer. The following sections discuss these changes to CONFIG.SYS.

Adding the COUNTRY Directive

The first step in making DOS go international is to issue the COUNTRY directive, which informs DOS how to customize itself for your location. When you issue the COUNTRY directive, DOS automatically changes the date and time formats to conform to the practice in the country specified. In addition, DOS can tell inquiring programs which characters to use for lists, currency symbols, the thousand separator, and decimal fractions.

Reminder:
COUNTRY's syntax depends on which version of DOS you are using.

The syntax for COUNTRY differs according to which version of DOS you use. For DOS V3.0 through V3.2, you use this syntax:

COUNTRY = nnn

nnn is a three-digit country code.

For DOS V3.3 and V4, you use the following:

COUNTRY = countrycode *codepage d:path\filename.ext*

countrycode is the mandatory three-digit code for the country. The default country code is that of the United States—001.

codepage is the code page number (see table 15.2). North American users of English should use the United States code page number (437). If you cannot find a code page number for your country, the Multilingual code page number (850) is probably your best choice.

Table 15.2
Country Codes

Country/Language	*Country Code*	*Keyboard Code*	*Existing Code Page*	*New Code Page*
Arabic	785	None	None	None
Australia	061	US	437	850
Belgium	032	BE	437	850
Canada (English)	001	US	437	850
Canada (French)	002	CF	863	850
Denmark	045	DK	865	850
Finland	358	SU	437	850
France	033	FR	437	850
Germany	049	GR	437	850
Israel	972	None	437	850
Italy	039	IT	437	850
Japan*	081	None	437	932
Korea*	082	None	437	934
Latin America	003	LA	437	850
Netherlands	031	NL	437	850
Norway	047	NO	865	850
PRC*	088	None	437	938
Portugal	351	PO	860	850
Spain	034	SP	437	850
Sweden	046	SV	437	850
Switzerland (French)	041	SF	437	850
Switzerland (German)	041	SG	437	850
Taiwan*	086	None	437	936
United Kingdom	044	UK	437	850
United States	001	US	437	850

*Added in DOS V4. Supported only on versions sold in Asia, for use on Asian hardware.

d:path\filename.ext is the path and file name for the file that contains country information. If you use the COUNTRY directive but do not provide a file name, DOS assumes that the information is in the COUNTRY.SYS file (which is supplied on the DOS Startup/Operating or Startup disk). COUNTRY.SYS is the only DOS-supplied file that you can specify as *filename.ext*.

Cue: *To skip the code page in the COUNTRY directive, insert two commas between the country code and the COUNTRY.SYS file.*

As you can see in the syntax, the code page and the country information file do not need to be specified. To skip the code page, use two commas between the country code and the COUNTRY.SYS file:

```
COUNTRY = 001,,C:\COUNTRY.SYS
```

The two commas instruct DOS to use the default code page. You also may omit the file name and supply either the country code or both the country code and code page, as in these examples:

```
COUNTRY = 001
```

```
COUNTRY = 001 437
```

Note that if you use the COUNTRY directive, DOS searches for the file COUNTRY.SYS in the root directory of the boot disk. If DOS cannot find the file, it displays the following message:

```
Bad or missing \COUNTRY.SYS
```

Reminder: *Always supply the full path name for COUNTRY.SYS.*

Although the full path name for your COUNTRY.SYS is not required, you should always supply it. Doing so prevents the possibility that DOS will be unable to locate the file.

Table 15.3 shows how DOS changes the display when you use various country codes.

DOS V3.3 and V4 make available to programs additional information about a country, including information about switching lowercase and uppercase letters and sorting characters' equivalents. DOS uses the country code and the code page, both of which affect the sorting order, to maintain information on sorting characters.

COUNTRY does have some limitations. One is that DOS does not automatically display its prompts in languages other than English. For that refinement, DOS messages must be rewritten. The modification of DOS messages is a technical subject beyond the scope of this text.

Adding DISPLAY.SYS

Reminder: *Using DISPLAY.SYS, you can switch code pages without restarting DOS.*

To display more than one code page or to print non-English language characters, you must use DISPLAY.SYS and PRINTER.SYS. Using DISPLAY.SYS in your CONFIG.SYS file enables you to switch code pages without restarting DOS. Using PRINTER.SYS enables you to download a font table to certain printers so that they can print non-English language and graphics characters.

Table 15.3
Effects of Country Codes on the Display

	US-English	*UK*	*Sweden*
Country code	1	44	46
Code page*	437	437	437
Currency symbol	$	£	SEK
Digits after decimal	2	2	2
separator in currency			
Thousand separator	,	,	.
Decimal separator	.	.	.
Data list separator	,	,	;
Date separator	-	-	-
Time separator	:	:	.
Time format	24-hour	24-hour	24-hour
Current date	Tue	Tue	Tue
	3-21-1989	21-03-1989	1989-03-21
Current time	16:24:39.01	16:24:39.01	16.24.39,01

*Although the country code currently supports 19 countries (23 with DOS V4), only five code pages are available.

The DISPLAY.SYS driver enables you to switch code pages on video adapters other than the Monochrome Display Adapter or the Color Graphics Adapter. The adapters that can be used effectively with DISPLAY.SYS are the PC Convertible LCD, the EGA, and the VGA. In DOS V4, if you do not specify the adapter type, DISPLAY.SYS checks your hardware to determine the type of active display.

To add DISPLAY.SYS to the CONFIG.SYS file, you use the following syntax:

DEVICE = *d:path***DISPLAY.SYS CON: = (type,** *hw_codepage,*
(added_codepages, subfonts))

d:path\ is the drive name and path to DISPLAY.SYS. If DISPLAY.SYS is not in the root directory of your boot disk, give the disk drive and path name.

CON: is the DOS device name for the console—the keyboard and display. The colon is optional.

The parameters of DISPLAY.SYS are enclosed in two sets of parentheses. The first parameter, the display **type**, can be any of the following:

MONO	Monochrome Display Adapter
CGA	Color Graphics Adapter

EGA	Enhanced Graphics Adapter or PS/2
LCD	Convertible LCD

The second parameter is *hw_codepage*, which stands for the hardware code page built into your adapter. The possible values are 437, 850, 860, 863, and 865—the five code pages that IBM provides with DOS V3.3 and V4. If you do not supply the *hw_codepage*, DISPLAY.SYS uses the code page given to the COUNTRY directive.

added_codepages is the maximum number of additional code pages the adapter can use. The range is from 0 (if you have an MDA or CGA) to 12 (if you are using the PC Convertible LCD, an EGA, or a VGA). *added_codepages* actually refers to the number of code pages you can prepare and use with the MODE command, which is discussed later in this chapter.

subfonts selects the dot resolution of the character: 9 by 14, 9 by 16, or 8 by 8. This number can range from 0 to 2, depending on the adapter. For the MDA and CGA, the default *subfonts* value is 0; for the LCD, the value is 1; and for the EGA or VGA, *subfonts* can be 1 or 2. For the EGA, DOS defaults to 2 *subfonts* if you omit this parameter.

The following is a sample DISPLAY.SYS directive for those who use an EGA or VGA and 437 as the hardware code page:

```
DEVICE = C:\SYS\DISPLAY.SYS CON = (EGA,437,1)
```

This command specifies an EGA adapter (EGA is used for both the EGA and VGA displays), 437 as the the starting code page, and one additional code page. This example assumes that the DISPLAY.SYS file is placed in a subdirectory called \SYS on drive C:. The following command line assumes that an EGA adapter is used and that the existing code page (that is, the default code page for the country where the computer is sold) is 863, for French-speaking Canada:

```
DEVICE = C:\SYS\DISPLAY.SYS CON = (EGA,850,2)
```

Notice that the existing code page is 863 but that the specified code page is 850. If your existing code page is not 437, always use 850 as the new code page and 2 as the value for additional code pages.

Note: If you use both ANSI.SYS and DISPLAY.SYS in your CONFIG.SYS file, the ANSI.SYS directive must appear before the DISPLAY.SYS directive. If the order is reversed, you lose the capability of ANSI.SYS.

Reminder: *Using PRINTER.SYS, you can print on certain printers those characters shown in the code page sets.*

Adding PRINTER.SYS

PRINTER.SYS enables the characters shown in the code page sets to be printed on certain printers. DOS V3.3 supports code pages on the following printers:

IBM ProPrinter Model 4201
IBM Quietwriter III Model 5202

DOS V4 adds support for these printers:

IBM ProPrinter Model 4202
IBM ProPrinter Model 4207
IBM ProPrinter Model 4208

To use code page switching with these printers, you must add the DEVICE = PRINTER.SYS line to your CONFIG.SYS file. The syntax for adding PRINTER.SYS is similar to that for adding DISPLAY.SYS:

DEVICE = *d:path***PRINTER.SYS LPTx: = (type,**
hw_codepage, added_codepages) ...

Or the following syntax will work:

DEVICE = *d:path***PRINTER.SYS LPTx: = (type,**
(hw_codepage1,hw_codepage2), added_codepages) ...

d:path\ is the drive name and path to PRINTER.SYS. If PRINTER.SYS is not in the root directory of your starting disk, supply the disk drive and path name.

LPTx: is the DOS device name for the line printer. The **x** is the number of the parallel printer: 1, 2, or 3. The colon is optional.

The parameters of PRINTER.SYS are enclosed in parentheses. The printer **type** is the printer's model number: for the ProPrinter, 4201 (with DOS V3.3) or 4201 and 4208 (with DOS V4); for the Quietwriter III, 5202. For DOS V4, the 4201 parameter is for the 4201 or 4202 ProPrinter; the 4208 parameter is for the 4207 or 4208 ProPrinter.

The second parameter, *hw_codepage*, stands for the hardware code page built into your printer. As with DISPLAY.SYS, the possible values are 437, 850, 860, 863, and 865.

There is a difference between the possible *hw_codepage* values for the two kinds of printers. ProPrinters have one font built into the device. This font is moved to the printer's RAM. When you change the code page for this printer, DOS overwrites the font that is in memory.

The Quietwriter III, however, uses hardware font cartridges. Each cartridge holds several code pages in the same type style. To use a particular code page, you must select the appropriate cartridge. The Quietwriter III cannot inform DOS when a cartridge has been changed; DOS must assume that the appropriate cartridge is in the printer.

The result is that you can specify one hardware code page for ProPrinters and two for the Quietwriter III. If, however, you specify two code pages for the Quietwriter III, you cannot prepare any code pages for the printer—DOS assumes that both code pages are fixed in hardware.

For ProPrinters, the value for *hw_codepage* follows the same guidelines as for DISPLAY.SYS. For the Quietwriter III, the value follows the same guidelines, with one exception: you may specify two hardware code pages, in which case DOS assumes that no code pages should be prepared for the Quietwriter III.

added_codepages is the maximum number of additional code pages that the printer can use. *added_codepages* can range from 0 to 12.

For ProPrinters, if you either specify 0 or omit *added_codepages*, you get the ProPrinter's built-in font. If you specify a number other than 0, the number is the maximum number of additional code pages the ProPrinter can use. Generally, use 1 if your hardware code page is 437; otherwise, use 2.

For the Quietwriter III, if you have specified two code pages, you must either omit *added_codepages* or specify a value of 0. The reason is that no code pages can be prepared if you have specified two hardware code pages.

An example of the directive to use with a ProPrinter is the following:

```
DEVICE = C:\SYS\PRINTER.SYS LPT1 = (4201, 437,1)
```

This directive calls for a ProPrinter connected to the first parallel port (LPT1), for 437 as the starting code page, and for one additional code page. As with DISPLAY.SYS, this example assumes that the PRINTER.SYS file is placed in a subdirectory called \SYS on drive C:. If PRINTER.SYS is in a different directory, change the disk drive and path information but leave all other information the same.

The following command assumes that a ProPrinter is used and that the existing code page is 863 (French-speaking Canada):

```
DEVICE = C:\SYS\PRINTER.SYS LPT1 = (4201,850,2)
```

Notice that the existing code page is 863 but that the specified code page is 850. Again, if your existing code page is not 437, always use 850 as the new code page, and use 2 as the value for additional code pages.

Now look at an example for the Quietwriter III, using a cartridge that holds both a 437 code page and an 850 code page:

```
DEVICE = C:\SYS\PRINTER.SYS LPT1 = (5202,(437,850),0)
```

Reminder:
If you specify two code pages, surround them with parentheses.

This directive follows the general form of the ProPrinter directive but includes the model number of the Quietwriter III (5202). Notice the parentheses that enclose the two code pages. If you give two code pages, you must surround them with parentheses. In addition, the number of added code pages is set to 0.

If you have more than one of these printers attached to your system, you can include that information on one directive line. For example, if the ProPrinter is connected to LPT1 and the Quietwriter III is connected to LPT2, you can use the following command:

```
DEVICE = C:\SYS\PRINTER.SYS LPT1 = (4201,437,2) LPT2 =
         (5202,(437,850),0)
```

You simply use a space after the information for LPT1 and before LPT2. DOS loads PRINTER.SYS and sets up both printers at the same time.

Changing the AUTOEXEC.BAT File

DOS V3.3 and V4 international language support depends not only on the COUNTRY directive and the two device drivers, but also on the following additional directives to set up and switch code pages for your display, printer, and keyboard:

NLSFUNC	Enables the CHCP command
CHCP	Activates (changes) code pages
KEYB	Loads and activates code pages for the keyboard
MODE	Loads and activates code pages for the display and printer

You can include these directives in the AUTOEXEC.BAT file. Table 15.4 summarizes the effects that these directives have on DOS, the console (the CON: device, including both the display and the keyboard), and the printer. You may want to refer to this table as you read the rest of the chapter.

Reminder: *You can include CHCP, NLSFUNC, KEYB, and MODE in the AUTOEXEC.BAT file.*

CHCP, NLSFUNC, KEYB, and MODE are closely linked. CHCP depends on NLSFUNC, and KEYB depends on MODE and CHCP. The following sections present these commands, explaining briefly what each command does, how to use it, and what steps must be performed before and after you use the command.

Supporting National Languages with NLSFUNC

NLSFUNC, the *national language support function*, is a DOS V3.3 program that performs two tasks. First, it hooks into DOS and provides programs with extended information about the specified country. Second, the program enables you to use the CHCP (change code page) command. If you will be switching code pages for any device except the keyboard, you need the NLSFUNC command. Fortunately, NLSFUNC is one of the easiest commands to use.

You use the following syntax for NLSFUNC:

*dc:pathc***NLSFUNC** *d:path**COUNTRY.SYS*

dc: is the name of the disk drive that holds the command, and *pathc*\ is the directory path to the command. The single parameter to NLSFUNC is the full file name to the COUNTRY.SYS file. To simplify the use of NLSFUNC, use the COUNTRY directive in your CONFIG.SYS file and give the full file name for the

Table 15.4
Effects of Various Commands and Device Drivers on DOS, the Console, and the Printer

Command/Device Driver	*DOS*	*Display*	*CON Keyboard*	*Printer*
DISPLAY.SYS	X	X	O	-
PRINTER.SYS	-	-	-	X
NLSFUNC	X	-	-	-
CHCP	X	X	X	X
MODE CODEPAGE PREPARE	-	X	O	X
MODE CODEPAGE SELECT	-	X	O	X
MODE CODEPAGE REFRESH	-	X	O	X
KEYB	-	O	X	-

Key: X Sets or changes
O Affects but does not set or change
- No effect

country information file. Also place the NLSFUNC command in your AUTOEXEC.BAT file. Certain applications programs may require that you run NLSFUNC. These programs use the new extended country information provided by DOS V3.3. Your program's documentation should state whether the program requires NLSFUNC.

Reminder: *Using NLSFUNC increases DOS's use of RAM by almost 2,700 bytes.*

Because NLSFUNC becomes part of DOS, using NLSFUNC increases DOS's use of RAM by almost 2,700 bytes. NLSFUNC remains in memory until you turn off the computer or restart DOS. You need to run NLSFUNC only once after you have started DOS. Running NLSFUNC a second time produces an error message.

Changing the Code Page with CHCP

CHCP, or *change code page*, was new with DOS V3.3 and is the simplest method of changing code pages. Because the CHCP command can affect the keyboard also, those using the KEYB program should be familiar with CHCP.

The syntax for CHCP is the following:

CHCP *codepage*

codepage—the new code page for DOS and all the devices to use—can be 437, 850, 860, 863, or 865.

Using CHCP sets the code page for DOS and establishes information about sorting and about switching lowercase and uppercase letters. CHCP also sets the code page for devices. If you have included DISPLAY.SYS in your CONFIG.SYS file, CHCP sets the code page for the display. If you have included PRINTER.SYS in CONFIG.SYS, CHCP sets the code page for the printer. If you have used the KEYB program, CHCP sets the keyboard's code page as well. Because of these capabilities, CHCP can be thought of as a system-wide program. If you have set up any device to use code page switching, CHCP will set the device.

Before you use CHCP, you must first be sure that you have performed the following actions:

- Issued the NLSFUNC command
- Given the COUNTRY directive in the CONFIG.SYS file
- Issued the necessary MODE CODEPAGE PREPARE commands

  (MODE is covered later in this chapter.)

You use the following command to set the code pages for DOS and all the devices to 437:

CHCP 437

You can determine the current code page that DOS is using by simply issuing the CHCP command without a code page number:

CHCP

Reminder: *To determine the current code page, issue the CHCP command without a code page number.*

Interestingly, CHCP can report the current code page for DOS even if you have not yet used NLSFUNC. CHCP does not increase the size of DOS and can be issued as often as you want.

Selecting a Language for the Keyboard with KEYB

The KEYB command is a program (or programs) that makes the keyboards of the PS/2, Personal Computer, and compatible systems multilingual. KEYB determines which characters can be typed on the keyboard.

Reminder: *KEYB makes the keyboards of the PS/2, Personal Computer, and compatible systems multilingual.*

KEYB for DOS V3.3 and V4 is quite different from KEYB for versions of DOS prior to V3.3. KEYB programs for DOS V3.2 and earlier versions should not be used with DOS V3.3 or V4. Each version is discussed separately in the following sections.

KEYB for DOS V3.2 and Earlier Versions

You use the following syntax for the KEYB command for DOS V3.2 and earlier versions:

*d:path***KEYxx**

d:path\ is the optional disk drive and path name to the KEYB programs. The **xx**, which can be any one of the keyboard codes in table 15.5, adapts the keyboard and the display to the character set of a particular language.

Table 15.5
Keyboard Codes for KEYBxx for V3.0–V3.2

Country	*Code*
France	FR
Germany	GR
Italy	IT
Spain	SP
United Kingdom	UK
United States	US

Reminder: *KEYBxx remains in memory until you reboot DOS or turn off the computer.*

The KEYBxx program takes about 2K of memory and becomes a resident program. KEYBxx remains in memory until you reboot DOS or turn off the computer. Once the program is loaded, the native language character set for the KEYBxx program becomes active.

KEYB for DOS V3.3 and V4

The KEYB command for V3.3 and V4 has a different syntax and set of requirements. KEYB alters the code page of the console (CON:, which includes both the display and keyboard). If you will be using keyboard layouts that need different code pages, you must "prepare" the console; that is, you must use the MODE CON CODEPAGE PREPARE command before you use KEYB.

You use the following syntax for KEYB V3.3:

*d:path***KEYB** *keyboardcode, codepage, d:path**filename.ext*

DOS V4 enables you to add one more parameter:

/ID:zzz

keyboardcode is the two-character keyboard code for your country (see table 15.5). If your country is not listed in table 15.5, use the keyboard code that meets your needs most closely. If you want to set the keyboard to use a different language, you must give a keyboard code.

codepage is the optional code page that you want to use with the keyboard. If you do not specify a code page, KEYB uses the default code page.

With DOS V4, you may use the */ID:zzz* parameter to select a specific keyboard layout for those countries that have available more than one Enhanced Keyboard. These are France (189, 120), Italy (141, 142), and the United Kingdom

(166, 168). Replace *zzz* with the keyboard ID code for the Enhanced Keyboard. The manual that came with your computer will let you know the layouts of the various Enhanced Keyboards so that you can select the one you want.

The final parameter of the KEYB command is optional. *d:path\filename.ext* is the full disk drive, path, and file name for KEYBOARD.SYS, the file that holds the information KEYB uses to build the various translation tables. KEYB places one restriction on keyboard codes and code pages: the keyboard code, the code page given to KEYB, and the console code page (as well as the keyboard ID number, if used) must all be compatible. Table 15.6 shows the allowable combinations for the first three parameters.

Table 15.6
Compatible Code Page and Keyboard Code Combinations for KEYB

Code Page	*Keyboard Codes*
437	FR, GR, IT, LA, NL, SP, SU, SV, UK, US
850	All
860	PO
863	CF
865	DK, NO

The reason for this requirement has to do with the way KEYB works. Basically, KEYB translates into ASCII characters the raw on-and-off codes produced by the keyboard hardware. The ASCII characters built by KEYB are determined by the keyboard code and code page used by KEYB and by the code page used by the console. KEYB builds a table of *keystroke codes to character translations* for each prepared console code page. The different combinations substitute the correct characters for each language. With each set of substitutions, some characters "disappear," and other characters appear.

However, KEYB can build these translation tables only for certain keyboard/code page combinations. Occasionally, graphics characters must be sacrificed for language characters. The characters to type for these languages appear in certain code pages but not others. Hence, using a keyboard layout with the wrong console code page means that you get the wrong characters on-screen. You type the "right" character, but you see the wrong character. For this reason, the keyboard code and code page that KEYB uses must agree with the console code page.

Reminder:
The keyboard code and code page used by KEYB must agree with the console code page.

Avoiding the following traps will help you use keyboard codes and code pages effectively:

- Do not give a keyboard code and code page that don't agree.
- If you are switching code pages, do not give a code page that is not in the list of code pages given to a MODE CON CODEPAGE PREPARE command.

- Do not give a keyboard code that does not agree with the console (CON) code page used in a CHCP command or a MODE CON CODEPAGE PREPARE SELECT command.
- Do not use KEYB with a keyboard code or a code page that conflicts with the current console code page. In other words, use MODE CON CODEPAGE SELECT or CHCP to switch the console code page before you use KEYB for a different code page. If you do not, KEYB will mutter a warning about incompatible combinations.

The manual for your computer and the DOS reference manual have a set of templates showing the different keyboard layouts. If you would like to experiment, you can load the KEYB programs and try typing some characters.

When any version of KEYB is loaded, you can switch easily between the U.S. character set and another language set. The key combination Ctrl-Alt-F1 switches to the U.S. character set. Ctrl-Alt-F2 switches back to the other language set.

The KEYB program stays in memory until you either reboot DOS or turn off the computer. You can use the KEYB program for DOS V3.3 as many times as you want. The first time you use KEYB, it increases the size of DOS by about 7,800 bytes. Using KEYB again does not increase the size of DOS.

If you have the KEYB program for DOS V3.2 or an earlier version, you should load only one KEYBxx program. In DOS V3.2, KEYB consumes about 2K of RAM. If you load a second KEYBxx, it assumes control, and the first KEYBxx program is deactivated. The first program, however, is trapped in memory and will continue to eat its 2K of space until you reboot DOS or turn off the computer.

Some computers are manufactured specifically for non-U.S. use. These computers have a different ROM BIOS and multilingual character set. Loading KEYB has no effect on such computers.

Changing the Code Page with MODE

If you want to use different code pages with your computer, you must prepare each code page for each device and then activate the code page. The MODE command prepares and selects code pages for each device.

Reminder:
The MODE command controls devices.

MODE is the command that controls devices. Before DOS V3.3, MODE consisted of four subcommands. In DOS V3.3, MODE has eight subcommands. The four additional subcommands, covered in Chapters 8 and 9, handle code page switching. (The remaining functions of MODE are covered in Chapter 9.) Fortunately, you do not need to use every MODE subcommand each time you start DOS. To switch code pages on a device, you must assemble the code pages for use. The MODE CODEPAGE PREPARE command loads code pages into the

computer's memory. The MODE CODEPAGE SELECT command downloads any needed code page to the device and tells that device which code page to use. MODE CODEPAGE SELECT also can tell the device to "forget" a software font and switch back to the font built into the device.

Because some fonts are placed in the device's RAM, you might turn the device off and lose the information. Or perhaps a stray program might wipe out the code page information. For these reasons, the MODE command has the capability of reminding the device of the font in use. This reminding is the function of MODE CODEPAGE REFRESH.

The MODE CODEPAGE /STATUS command rounds out MODE, displaying the number of the available code pages for a device and indicating which code page is active.

DOS provides some abbreviations for parts of the command. You can use the abbreviations listed in table 15.7 when you use the MODE CODEPAGE command.

Table 15.7
MODE CODEPAGE Abbreviations

Word	*Shorthand*
CODEPAGE	CP
PREPARE	PREP
SELECT	SEL
REFRESH	REF
STATUS	STA

Defining Code Pages with MODE CODEPAGE PREPARE

The first step in using switchable code pages is to use the MODE CODEPAGE PREPARE command to load the disk-based code pages into the computer's memory. The syntax for MODE CODEPAGE PREPARE is the most complex of any MODE subcommand, but the command can be tamed. Here is the command syntax:

Reminder: *Use MODE CODEPAGE PREPARE to load the disk-based code pages into the computer's memory.*

*dc:pathc***MODE device CODEPAGE PREPARE = ((code_page, ...)**
*d:path***filename.ext**)

dc:pathc\ is the optional disk drive and directory path to the MODE command. **device** is the name of the device that MODE uses. You must specify the device, which can be any of the following:

CON: Console

LPT1: First parallel printer

LPT2:	Second parallel printer
LPT3:	Third parallel printer
PRN:	The current system printer, usually a synonym for LPT1:

As with all device names, the colon is optional. For each specified device, a driver must have been loaded through the CONFIG.SYS file. To use CON in this command, you must include DISPLAY.SYS in CONFIG.SYS; to use the remaining devices, you must include PRINTER.SYS in CONFIG.SYS.

The **code_page** numbers must be enclosed within a set of parentheses. If you need more than one code page, separate the code page numbers with a comma. The ellipsis (...) represents the other optional code pages. Remember to give a closing parenthesis after the final code page.

For example, to use 850 as a code page, you use the following form:

= (850)

To use the 437 and 850 code pages, you use this form:

= (437, 850)

Spaces are optional here.

Reminder:
Do not give a hardware code page number to MODE CODEPAGE PREPARE.

Avoid giving a hardware code page number to MODE CODEPAGE PREPARE. In other words, remember the code page that you gave to DISPLAY.SYS or PRINTER.SYS. Do not give MODE CODEPAGE PREPARE the same code page number. If, for example, you used the following line in your CONFIG.SYS file, do not give 437 to MODE CODEPAGE PREPARE:

DEVICE = C:\SYS\DISPLAY.SYS CON = (EGA, 437, 2, 2)

The driver already knows that the 437 code page is built into the adapter. Giving 437 to MODE simply causes MODE to reload the 437 code page, wasting memory that other fonts could use. You also force the computer to download the font each time the 437 code page is selected—a minor, but avoidable, waste of time.

The final information you must give to MODE CODEPAGE PREPARE is the drive name, path name, and file name for the file that holds the font information for the device. The general form of the file name is **devicename.CPI**. The **devicename** can be any of the following:

EGA	EGA and VGA display adapters
LCD	PC Convertible LCD display
4201	IBM ProPrinter
5202	IBM Quietwriter III

The CPI extension is for *code page information*.

The EGA.CPI and LCD.CPI files, of course, are used with the CON device. The 4201.CPI and 5202.CPI files are used with the PRN or LPT devices.

If the code page information file is not on the current disk or in the current directory, you must provide the needed information. The full file name is mandatory. Because all DOS-included code page files have an extension of CPI, you usually will give both the root name and the extension.

Note that you cannot prepare more code pages than the number you gave to DISPLAY.SYS or PRINTER.SYS for *added_codepages*. If you need more code pages, you must edit the DISPLAY.SYS or PRINTER.SYS line in your CONFIG.SYS file and then restart DOS.

Remember that if you gave two hardware code pages for the Quietwriter III, you must have given a value of 0 for *added_codepages*. DOS assumes that the two code pages are in the printer. In this case, you do not need to use MODE CODEPAGE PREPARE on the Quietwriter III.

If you issue the MODE CODEPAGE PREPARE command more than once, you can skip the code pages that you do not want to change. For example, you use the following MODE command to load the 850, 860, and 863 code pages:

```
MODE CON CODEPAGE PREPARE = ((850, 860, 863) C:\EGA.CPI)
```

If you later want to use the 850 and 863 code pages, but you want the 865 code page instead of the 860 code page, you simply drop the number and insert a comma for the unchanged code pages. To change the 860 code page to the 865, you use the following command:

```
MODE CON CODEPAGE PREPARE = ((,865,) C:\EGA.CPI)
```

A more common example would be first using this command to load into memory the Multilingual code page:

```
MODE CON CODEPAGE PREPARE = ((850) C:\EGA.CPI)
```

Later, you might want to add another code page to the list. For instance, to add the 860 code page, you would use this command:

```
MODE CON CODEPAGE PREPARE = ((,860) C:\EGA.CPI)
```

Now either the 850 or 860 code page can be used. Remember that the value for *added_codepages* given to DISPLAY.SYS must be at least 2 for this example to work and 3 for the preceding example to work.

Suppose that you give an incorrect code page number or give the name of an existing file that is not a code page information file. DOS displays the message `Font file contents invalid`. The code page at that position in the syntax line is removed from the list. Using the preceding example (which has the 850 and 860 code pages prepared), assume that you issue this command:

MODE CON CODEPAGE PREPARE = ((,836) C:\EGA.CPI)

DOS tells you that the code page file does not have the given font. The code page 836 cannot replace the code page in the second position, which was 860. However, 860 is removed from memory. To use 860, you must reload it with the MODE CODEPAGE PREPARE command.

Remember that the commands can be abbreviated, as in the following examples:

MODE CON CP PREP = (850, 863) C:\EGA.CPI

MODE LPT1 CP PREP = (860, 863) C:\4201.CPI

Activating Code Pages with MODE CODEPAGE SELECT

After code pages are prepared, they can be used. One command to activate the code pages is MODE CODEPAGE SELECT. (The other is CHCP.)

Cue:
If you do not want the same code page active on all devices, use MODE CODEPAGE SELECT rather than CHCP to activate code pages.

CHCP is the preferred command for activating code pages. You can use CHCP to activate the code pages for DOS and all the devices at one time. CHCP is also easier to use than MODE CODEPAGE SELECT. Occasionally, however, you may not want the same code page active on all devices. Then you must use MODE CODEPAGE SELECT.

Suppose that you have edited a document with code page 437 and have started printing the document in the background. Now you want to edit another document. That document was created with the 850 code page. You need to use the 860 code page for the console (the display and keyboard).

If you use CHCP in this instance, you affect both the display and the printer. Since you do not want that, your only choice is to use MODE CODEPAGE SELECT to switch fonts on the display.

You use the following syntax for MODE CODEPAGE SELECT:

*dc:pathc***MODE device CODEPAGE SELECT = codepage**

dc:pathc\ is the optional drive and directory path to the MODE command. **device** is the name of the device whose code page will be activated by MODE. **device** can be CON, LPTx, or PRN. **codepage** is the code page the device will use. **codepage** must be either the hardware code page given to the appropriate device driver or one of the code pages given to the MODE CODEPAGE PREPARE command. One, and only one, **codepage** can be given.

Using the MODE CODEPAGE SELECT command downloads the appropriate font to the device and activates the font. The device uses that font until you issue another MODE CODEPAGE SELECT or CHCP command, or until you turn off the device or restart DOS.

Ordinarily, you use one of these forms for MODE CODEPAGE SELECT:

MODE CON CP SEL = codepage

MODE LPT1 CP SEL = codepage

In the document example, you would enter the following:

MODE CON CP SEL = 860

Refreshing a Device's Memory with MODE CODEPAGE REFRESH

Reminder:
MODE CODEPAGE REFRESH reminds the device which font it should use.

MODE CODEPAGE REFRESH reminds the device which font the device should be using. Generally, the command is not needed on the display, unless a program has affected the screen in an undesirable manner. The command is used more frequently for the printer. If you turn the printer off and on again, the software code page is lost. Thus, DOS can lose track of the code page that the printer is using. The printer and DOS must then be resynchronized.

The MODE CODEPAGE REFRESH command reloads and activates any needed font. You use the following syntax for MODE CODEPAGE REFRESH:

*dc:pathc***MODE device CODEPAGE REFRESH**

device is the name of the device whose code page should be reestablished. Do not give a code page number. MODE uses the last code page used in the MODE CODEPAGE SELECT command.

To use MODE CODEPAGE REFRESH, you must have used MODE CODEPAGE PREPARE and either MODE CODEPAGE SELECT for the device, or the CHCP command. You can issue MODE CODEPAGE REFRESH as often as needed. The following are examples of the shorthand form of MODE CODEPAGE REFRESH:

MODE LPT1 CP REF

MODE CON CP REF

Checking Code Pages with MODE /STATUS

Reminder:
MODE /STATUS displays code page information about a device.

You can use the MODE /STATUS command to display code page information about a device. MODE /STATUS lists the following information:

- The number of the active code page, if any
- The number of the hardware code page
- The numbers of any prepared code pages

You use the following syntax for MODE /STATUS:

*dc:pathc***MODE device CODEPAGE /STATUS**

device is the name of the device whose code page information you want displayed. Notice that this command is a switch. You must give the switch character (the slash), followed by the word **STATUS**.

The next examples display the output of the MODE /STATUS command for the console. DOS was started with the following command in the CONFIG.SYS file:

```
DEVICE = C:\SYS\DISPLAY.SYS CON = (EGA, 437, 4)
```

The DISPLAY.SYS file has been set up for an EGA or VGA display (EGA), a hardware code page of 437, and four additional code pages. The first two of the following commands establish two of the four code pages. The third command requests the code page status from DOS.

```
C>MODE CON CODEPAGE PREP = (850, 860) C:\EGA.CPI
C>MODE CON CODEPAGE SELECT = 850
C>MODE CON CODEPAGE /STATUS
```

DOS issues the following status report:

```
hardware codepages:
   Codepage 437
prepared codepages:
   Codepage 850
   Codepage 860
   Codepage not prepared
   Codepage not prepared
MODE Status Codepage function completed
C>
```

The first MODE command sets up the console to use the 850 and 860 code pages, in addition to the 437 code page built into the display adapter. The second MODE command activates the 850 code page.

The first line of the MODE /STATUS command's report shows the active code page. The next two lines give the hardware code page based on the DISPLAY.SYS or PRINTER.SYS line in the CONFIG.SYS file. The lines that follow show the software code pages; the number of lines is based on the *added_codepages* number given to DISPLAY.SYS or PRINTER.SYS. If a code page is prepared, you will see the number listed. The message `Codepage not prepared` indicates that additional areas for code pages are available.

You can use MODE /STATUS at any time. However, you get an error message if the appropriate device driver has not been installed for the device. For the console, you can use the following abbreviated form for MODE /STATUS:

```
MODE CON CP /STA
```

For the first line printer, you can use this form:

```
MODE LPT1 CP /STA
```

Solving Display Problems with GRAFTABL

A minor problem occurs in displaying non-English characters on the IBM Color Graphics Adapter (CGA). When you attempt to display characters in the ASCII range of 128 to 255 while in graphics mode, the characters are almost unreadable. Unfortunately, the non-English language characters are in the range of 128 to 255. This problem affects only the CGA. And the problem occurs only in medium-resolution, 320-by-200, 4-color mode—not in normal alphanumeric text mode or in high-resolution, 640-by-200, 2-color mode.

To get a legible display in medium-resolution graphics mode, use the GRAFTABL command. GRAFTABL loads the alternative character set into the computer's memory and then directs the CGA to use this set when in graphics mode. The ASCII characters in the range of 128 to 255 are thus displayed legibly.

Reminder: *GRAFTABL provides a legible display in medium-resolution graphics mode.*

You use the following syntax for GRAFTABL:

*dc:pathc***GRAFTABL**

The GRAFTABL command uses about 1,200 bytes of RAM. This memory remains trapped until you either restart DOS or turn off the computer. Running GRAFTABL more than once simply wastes 1,200 bytes of memory each time you run the command. If a program that you frequently run requires GRAFTABL, add the command to your AUTOEXEC.BAT file.

With DOS V4, GRAFTABL was enhanced to support code page 850, the Multilingual code page.

Using SELECT To Set Country Information

The utility program SELECT was extended to enable you to install DOS V4 on a floppy disk or on your hard disk. Before DOS V4, however, SELECT existed as a program with more limited capabilities.

In DOS V3.x, SELECT is used solely to automate the selection of the country code and keyboard program. SELECT for versions of DOS before V3.2 destroys any previously established CONFIG.SYS and AUTOEXEC.BAT files. SELECT for V3.2 and V3.3 runs FORMAT and destroys all the files on the disk! If you have already set up your disk, you should simply edit your existing CONFIG.SYS and AUTOEXEC.BAT files, instead of running SELECT.

If you are using SELECT with floppy disks, you need the correct blank disk for your disk drive. If you have a 1.2M disk drive, you need a 1.2M floppy disk. If you have a 1.44M disk drive, you need a 1.44M floppy disk. Using a disk with the wrong capacity can cause SELECT to fail.

SELECT for DOS V3.2 and V3.3

Caution: *Using SELECT for DOS V3.2 and V3.3 on your hard disk destroys all information on the disk.*

SELECT for DOS V3.2 and V3.3 can work on floppy disks or hard disks. This version of SELECT runs the FORMAT command. If you use SELECT on your hard disk, you will destroy all information on the disk. For this reason, you should avoid running SELECT unless the hard disk has no files. You will gain more by creating or editing your CONFIG.SYS and AUTOEXEC.BAT files than by copying everything back to your hard disk after FORMAT has run.

You use the following syntax for SELECT for V3.2 and V3.3:

*dc:pathc***SELECT** *ds: dd:**pathd* **countrycode keyboardcode**

ds: is the name of the disk drive that holds the source disk and should hold the DOS Startup or Startup/Operating disk. *ds:* must be a floppy disk drive (either A: or B:). If you do not specify the source disk drive, SELECT uses drive A:.

dd: is the optional destination disk, the disk to be set up. The destination disk must be different from the source disk. If you do not give *dd:* and the current disk is not the source disk, SELECT uses drive B:. If you do not give *dd:* and the current disk is the source disk, SELECT displays an error message and aborts. To use SELECT on a hard disk, you must specify the hard disk's drive name.

pathd is the path to the directory to which SELECT copies the DOS files. You must use an absolute path name—that is, you must start with the root directory (\); the . and .. names are not allowed.

If you do not give *pathd*, DOS copies the files to the root directory of *dd:*. If the destination is a floppy disk, you generally can omit *pathd*. Because of the limited storage space, do not use a destination path if the destination is a 360K disk. If the destination is a hard disk, you should give *pathd*. A good name is either BIN or DOS.

countrycode is the three-digit code for your country, and **keyboardcode** is the two-character code for your country.

SELECT runs FORMAT to prepare a hard disk. FORMAT will ask you for a volume label. Putting volume labels on hard disks is a good processing practice. Because FORMAT requests the existing volume label before formatting a hard disk, inadvertently formatting a hard disk that has a volume label is almost impossible.

After FORMAT has run, SELECT runs XCOPY to transfer the files from the boot disk to the destination disk or hard disk. If you specified a destination path (*pathd*), XCOPY copies the files to the directory. SELECT then creates any needed directories. If you did not specify a destination path, XCOPY copies the files to the root directory of the destination disk.

Finally, SELECT creates a CONFIG.SYS file and an AUTOEXEC.BAT file. The CONFIG.SYS file for SELECT V3.2 and V3.3 holds one of the following lines:

```
COUNTRY = countrycode,codepage
COUNTRY = countrycode,codepage,d:path\COUNTRY.SYS
```

Either of these lines sets up the COUNTRY directive. The correct code page is based on the country code you gave to SELECT. The first line is used if you do not give a destination path. The second line, which has the full file name for COUNTRY.SYS, is used if you give a destination path. *d:path*\ is the destination drive and path name.

If the destination is a floppy disk, the AUTOEXEC.BAT file for V3.2 and V3.3 holds the following lines:

```
PATH \;\path
KEYB keycode codepage
ECHO OFF
DATE
TIME
VER
```

If the destination is a hard disk, AUTOEXEC.BAT holds these lines:

```
PATH d:\;d:\path
KEYB keycode codepage
ECHO OFF
DATE
TIME
VER
```

d: is the name of the destination disk drive. If you gave a destination path to SELECT, the first line substitutes that destination path for the path variable. If you did not give a destination path to SELECT, the second directory path is dropped. The line becomes either PATH \ or PATH d:\.

The AUTOEXEC.BAT file runs the KEYB program with the keyboard code you gave and inserts the compatible code page for the keyboard code. The batch file then turns off ECHO, asks for the date and time, and displays the DOS sign-on message. The last three commands approximate what DOS does if no AUTOEXEC.BAT file is on the disk.

If you run SELECT on a floppy disk, the disk you produce should be your new DOS working copy. If you run SELECT on a hard disk, your DOS files are copied to the hard disk. Put any additional CONFIG.SYS or AUTOEXEC.BAT command lines you want into these files.

A Sample COUNTRY Setup

In the examples presented here, assume that you have a hard disk and that all the DOS program files are copied to a subdirectory called C:\DOS. Assume also that all SYS-extension files have been copied to C:\DOS\SYS. If this is not true, change the disk drive and path name to match your system. Floppy disk users should omit the path name and use A: in place of C:.

The examples are for DOS V3.3 and V4 only. Consult your DOS manual for the correct setup if you are using an earlier version of DOS.

Adding International Directives to CONFIG.SYS

To use the international features of DOS, start by adding the following line to your CONFIG.SYS file:

```
COUNTRY = 001, 437, C:\DOS\SYS\COUNTRY.SYS
```

If you have a PC Convertible LCD display, an EGA, or a VGA, use the following DEVICE = DISPLAY.SYS line:

```
DEVICE = C:\DOS\SYS\DISPLAY.SYS CON = (EGA, 437, 1)
```

This line sets up the display for the 437 code page and one additional code page.

For a Monochrome Display Adapter, use this line:

```
DEVICE = C:\DOS\SYS\DISPLAY.SYS CON = (MONO, 437, 0)
```

Cue:
For a Monochrome Display Adapter, the number of additional code pages is always 0.

Remember that the number of additional code pages for a Monochrome Display Adapter is always 0.

If you use ANSI.SYS, the DRIVER.SYS line must appear after the ANSI.SYS line. Reversing this order causes you to lose the features that ANSI.SYS offers.

If you have either the IBM ProPrinter or the Quietwriter III, you should use PRINTER.SYS. For the ProPrinter, use the following line:

```
DEVICE = C:\DOS\SYS\PRINTER.SYS LPT1 = (4201, 437, 1)
```

This line sets up the ProPrinter for the first parallel port (LPT1), the 437 code page, and one additional code page.

In the preceding lines, if 437 were not your existing code page, you would use the existing code page and specify 2 instead of 1 for the additional code pages to be used. For example, the DISPLAY.SYS line would look like this:

```
DEVICE = C:\DOS\SYS\DISPLAY.SYS CON = (EGA, 863, 2)
```

The code page 863 for French-speaking Canada is built into the adapter, and two additional code pages are to be used.

Remember that the CONFIG.SYS file must be in the root directory of the boot floppy disk or hard disk and that you must restart DOS before the change takes place.

Adding International Directives to AUTOEXEC.BAT

Add the following lines to the AUTOEXEC.BAT file of your start-up disk. If the start-up disk is a hard disk, place these additional commands after the line that holds your PATH command. When the commands occur after PATH, DOS automatically finds the commands.

The line to add is simply this:

NLSFUNC

The next step is to use MODE CODEPAGE PREPARE on each device. If DISPLAY.SYS is installed, use the MODE CON CODEPAGE PREPARE command (abbreviated here):

MODE CON CP PREP = (850) C:\DOS\EGA.CPI

This command prepares the 850 code page for use with EGA/VGA-type displays. If your hardware code page is 437, use 850 as the prepared code page. If your hardware code page is not 437, use 850 and the existing code page:

MODE CON CP PREP = (850, 863) C:\DOS\EGA.CPI

This command adds the code page for French-speaking Canada to the Multilingual code page.

If PRINTER.SYS is installed, use the MODE LPTx CODEPAGE PREPARE command. The line for the ProPrinter is the following:

MODE LPT1 CP PREP = (850) C:\DOS\4201.CPI

This line prepares the 850 code page for use with the ProPrinter Model 4201. If the hardware code page is not 437, you prepare the 850 but add another code page:

MODE LPT1 CP PREP = (850, 863) C:\DOS\4201.CPI

This line adds the code page for French-speaking Canada to the Multilingual code page.

If you gave two hardware code pages for the Quietwriter III, you do not need to issue the MODE CODEPAGE PREPARE command for this printer. Otherwise,

follow the same form for the ProPrinter, but use 5202.CPI for the code page information file name.

After you add the commands to prepare the code pages, use the CHCP command to activate the pages. You add the following line to AUTOEXEC.BAT:

```
CHCP codepage
```

The command you type into your AUTOEXEC.BAT is this:

```
CHCP 437
```

437 is the existing code page.

Finally, add a line to enable the switching of your keyboard. For this example, use the following line:

```
KEYB US,,C:\DOS\SYS\KEYBOARD.SYS
```

If you use KEYB frequently on the hard disk and follow the suggestion to copy the file to the root directory, add this command to the AUTOEXEC.BAT file instead:

```
KEYB US,,C:\KEYBOARD.SYS
```

If you have a Color Graphics Adapter and use medium-resolution graphics mode frequently, add the following line:

```
GRAFTABL
```

Now restart DOS and watch your display. If you see any error messages, check the lines in your CONFIG.SYS and AUTOEXEC.BAT files. You may have misspelled a command or forgotten to give the correct file name. Correct the mistake and try restarting DOS.

Summary

In this chapter, you learned the following important points:

- Code pages are font sets. Some code pages are built into hardware, and some are loadable from a file.
- You use the CONFIG.SYS COUNTRY directive to tell DOS which international country code and which code page to use.
- DOS adapts the format for the date and time depending on the country code.
- The DISPLAY.SYS device driver enables code page switching on some video displays.

- The PRINTER.SYS device driver enables code page switching on some printers.
- The NLSFUNC command provides additional functions that make DOS go international and enables the CHCP command to be used.
- The CHCP command switches code pages for DOS and all devices at one time.
- The MODE CODEPAGE subcommands load, activate, reselect, and display selected code pages.
- The KEYB command alters the keyboard for different languages.
- The GRAFTABL command enables you to display legibly non-English language characters and certain graphics characters when you use the Color Graphics Adapter in medium-resolution mode.

Chapter 16 introduces and explains DOS V4.0's DOS Shell.

16

Using the DOS Shell

The DOS Shell is a user interface that replaces the DOS command line with easy-to-use and easy-to-understand menus. If you don't understand any aspect of the Shell, you can get help by pressing a function key.

With Microsoft Windows taking the industry by storm and OS/2's Presentation Manager just around the corner, DOS V4's Shell has been undeservedly ignored. The DOS Shell has several commendable features: you don't need a graphics monitor, you can use your mouse, you don't need much memory, and the Shell comes free with DOS V4.

Moreover, learning to use the DOS Shell can serve as an introduction to more complex systems like Windows. You can use the Shell to learn to install programs, manage the file system, and use pull-down menus; you can use many of these concepts with Windows or another shell.

Even if you don't intend to use the Shell yourself, you probably will encounter it at one time or another, so you should learn to use it. As a standard part of the latest version of DOS, the Shell is likely to see widespread use despite current neglect. This chapter explains what you need to know if you encounter the Shell when you sit down at a colleague's computer.

Getting To Know the Shell Menu Tree

The DOS Shell has a series of menus arranged in branching, tree fashion. The topmost menu provides a list of functions from which you may choose. These choices may include DOS commands, utility programs, or submenus with lists of selections of their own.

The Shell main menu is called the Start Programs/Main Group menu. The main menu is the jumping-off point to each of the other DOS Shell functions and submenus. From this menu, you can load programs or access additional menus (see fig. 16.1).

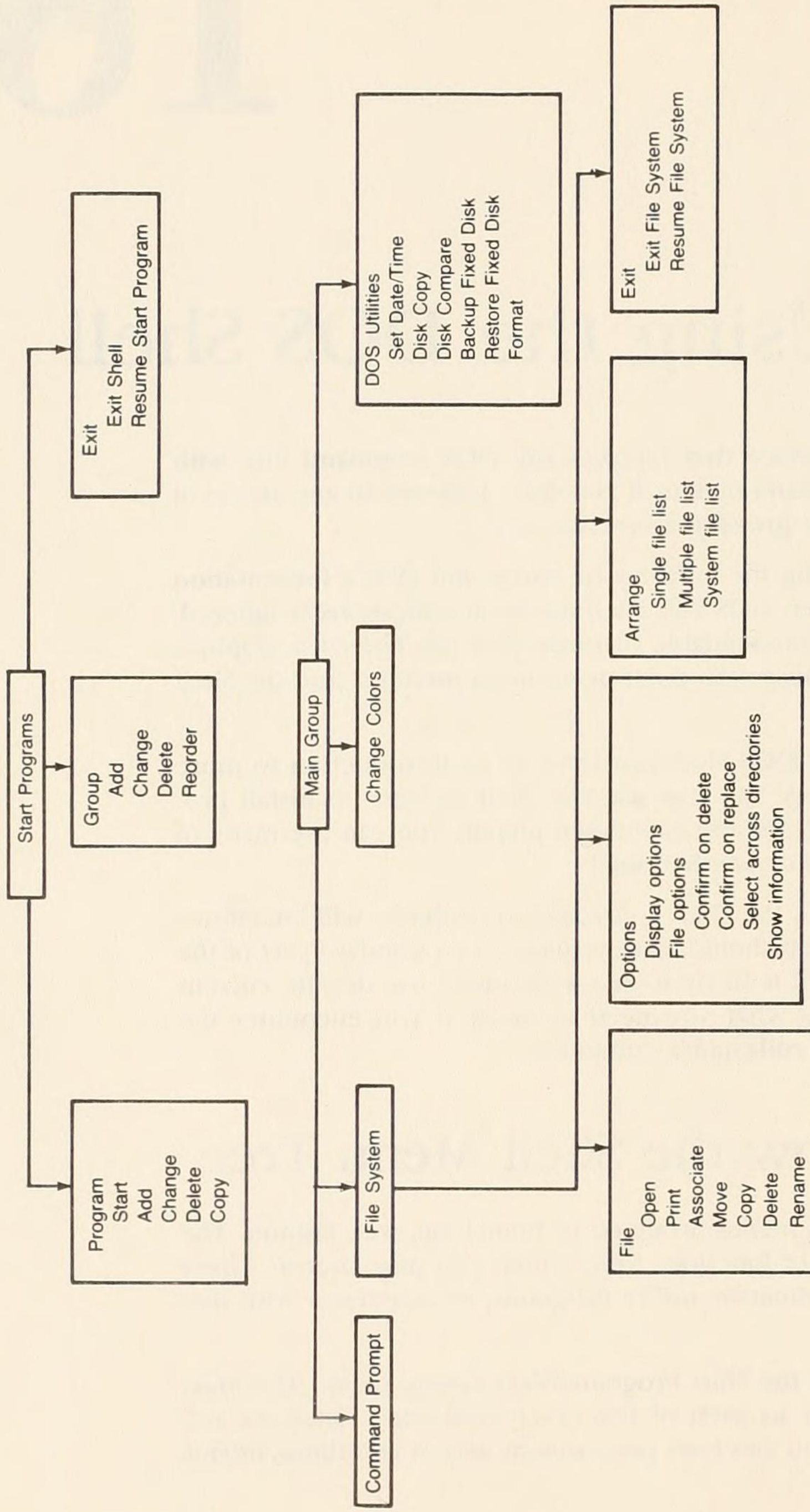


Fig. 16.1

Menu structure of the DOS Shell.

If you have any familiarity with menu trees, you will notice that the DOS Shell menu tree is odd: it is "split" into two sections that aren't directly connected. This is done so that you can do two things with DOS Shell menus. First, you can access various programs and utilities provided by the Shell. These functions include DOS commands, your own applications programs, and special DOS Shell utilities such as the File System or the Change Colors functions. The functions are accessed through the Main Group menu and the submenus beneath it.

Cue: *The DOS Shell menu tree is split so that you can change the Shell itself as well as access programs and utilities.*

The second thing you can do with the DOS Shell is to modify the Shell itself. That is, you can add programs, move programs to other parts of the menu, or modify how the programs are started by the DOS Shell. You also can add new groups of programs (submenus) to the tree or change the order in which groups of programs appear in the menus. The final thing you can do to the Shell is disable it—or leave it entirely. That is, you can exit from the DOS Shell and return to the DOS prompt. The Shell is unloaded from memory and is no longer available to you until you type **DOSSHELL** again.

NOTE: You also can exit from the Shell temporarily by using the Command Prompt option from the Main Group menu or by pressing Shift-F9. If you exit from the Shell this way, the Shell remains in memory and can be recalled by typing **EXIT** at the DOS prompt.

To differentiate between the two types of functions, DOS V4 sets up two different menus—Main Group and Start Programs. You can switch back and forth between these menus at any time by pressing F10. The Start Programs action bar contains the Program, Group, and Exit options which enable you to start or modify programs or groups of programs. The Main Group menu enables you to access any of the programs listed; you can also select one of the four options on this menu to display the associated submenu. Each of these menus is described later in this chapter.

Cue: *Press F10 to switch between the Start Programs "action bar" menu and the Main Group menu.*

Understanding the Shell Display Modes

The Shell can be displayed in one of four modes, depending on the type of monitor you have. Figure 16.2 shows the main menu in 25-line color graphics mode. This mode is used by most DOS V4 users who have Enhanced Graphics Adapter (EGA) or Video Graphics Adapter (VGA) color monitors. If you have a VGA or EGA monochrome monitor, your screen also looks like the one in figure 16.2. The mouse cursor (if you have a mouse) is displayed as an angled arrow.

The DOS Shell also operates in text mode (see fig. 16.3). If you have a monochrome or Color Graphics Adapter (CGA) monitor, this is the mode for you. You can set up your system to display in text mode even if you have a VGA or EGA monitor, but you gain no advantages. The mouse cursor is displayed as a graphics box in text mode.

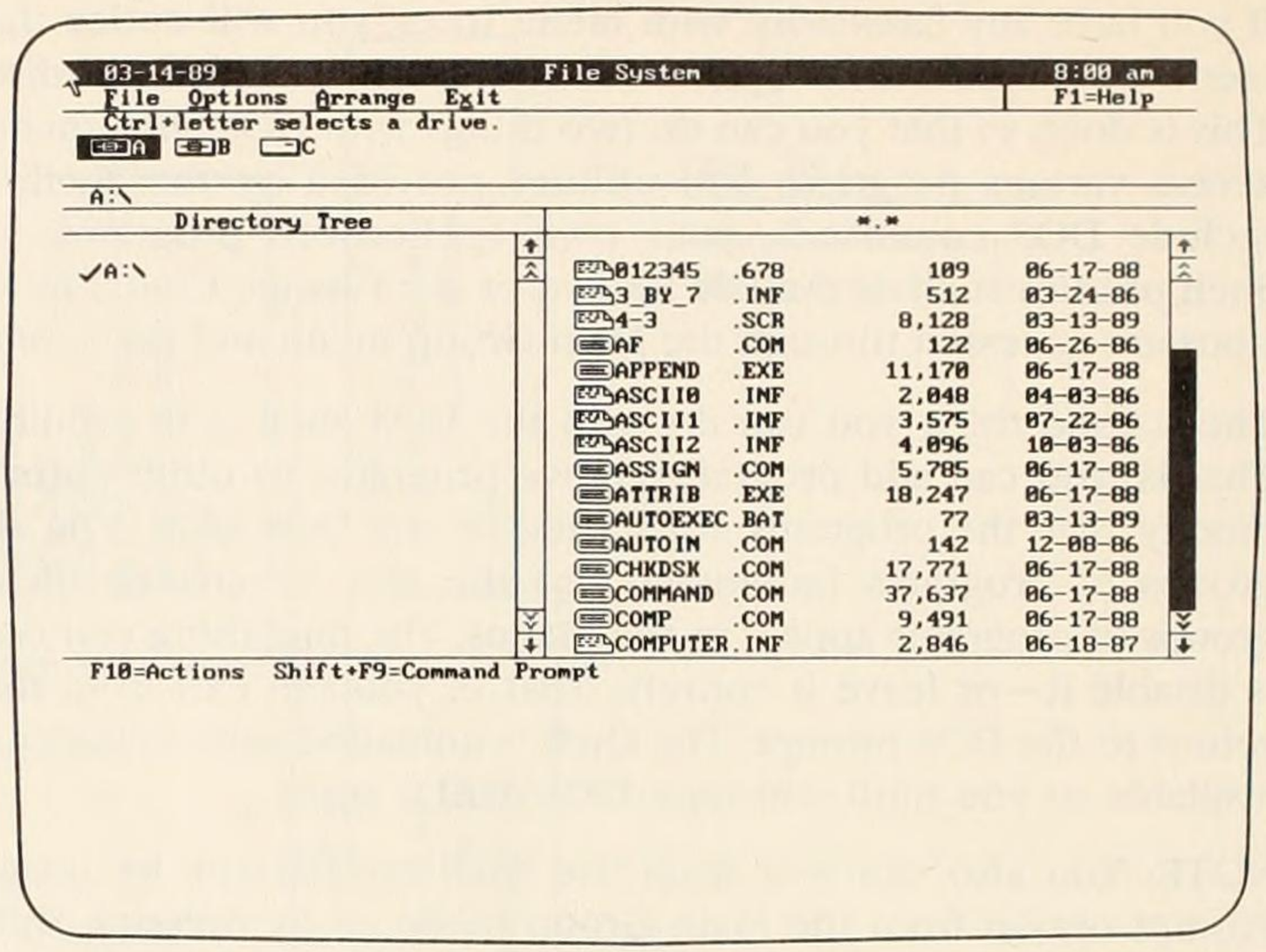


Fig. 16.2

The DOS Shell menu in 25-line color graphics mode.

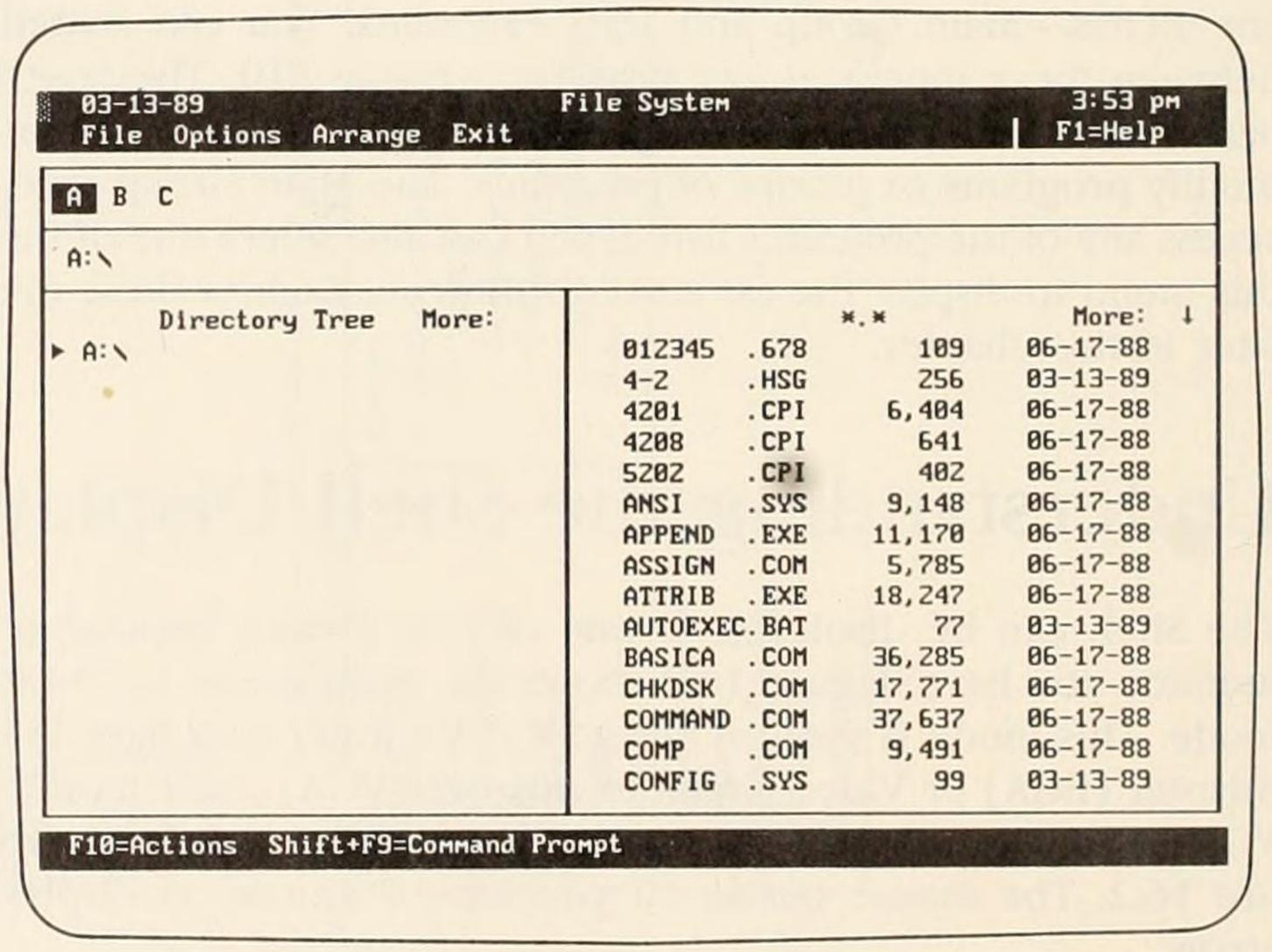


Fig. 16.3

The DOS Shell menu in text mode.

Third and fourth modes are available for use with VGA monitors: 30-line color and black-and-white graphics modes (see fig. 16.4). These modes have the advantage of displaying five extra lines on-screen. The text is slightly smaller

and may be more difficult to read on some monitors. With a good-quality monitor, however, this mode is worth using. The chief advantage comes when you display directories, because an additional five file names can be shown on a single screen without scrolling.

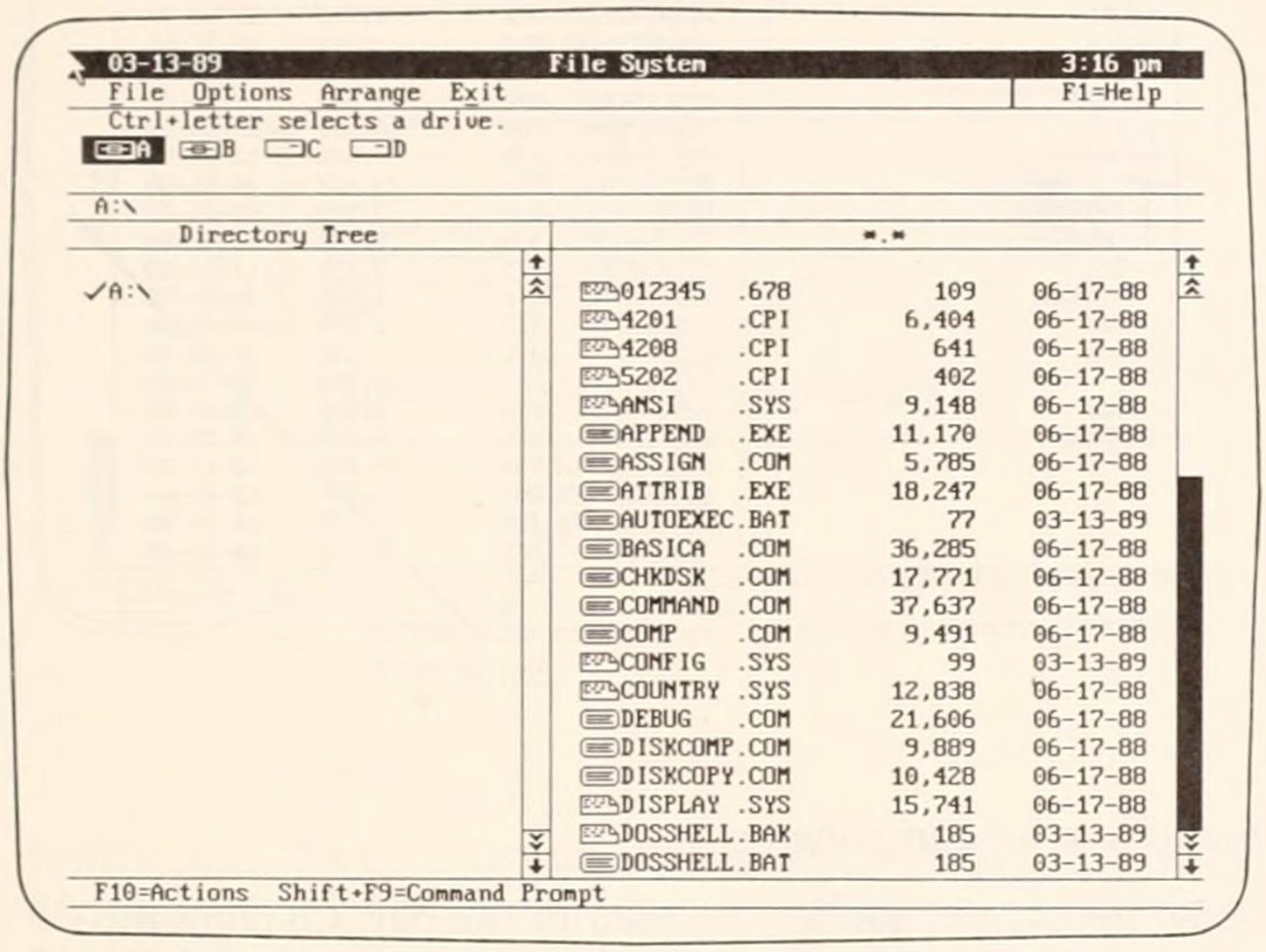


Fig. 16.4

The DOS Shell menu in 30-line black-and-white graphics mode.

Scrolling Text

Before you read about the Start Programs and Main Group menus, you should know how to move the cursor around the screen. You can use the mouse or the arrow keys to move the pointer around the Shell menus. You also know how to click the mouse button with the cursor inside a menu box. Here is another mouse trick to practice—scrolling text. Even if you do not have a mouse, you need to know how to scroll text.

Cue:
Use PgUp, PgDn, and the left and right arrows to scroll text in windows.

Many of the boxes and screens displayed by the DOS Shell include text. This text may be too long to show in the space allotted. If you don't have a mouse, you can see more text by pressing the down-arrow or up-arrow key to scroll the "window" to a new position. The PgUp and PgDn keys can also be used.

If you have a mouse, you know it can move the pointer beyond the boundaries of the window. You may not want to take your hand from the mouse to use the keyboard controls to scroll. If you use a mouse in text mode, however, you must use the keyboard controls to scroll. Graphics-mode mouse users, on the other hand, can use the special scroll bars to scroll text (see fig. 16.5).

Cue:
If you have a mouse in graphics mode, use the scroll bars to scroll quickly.

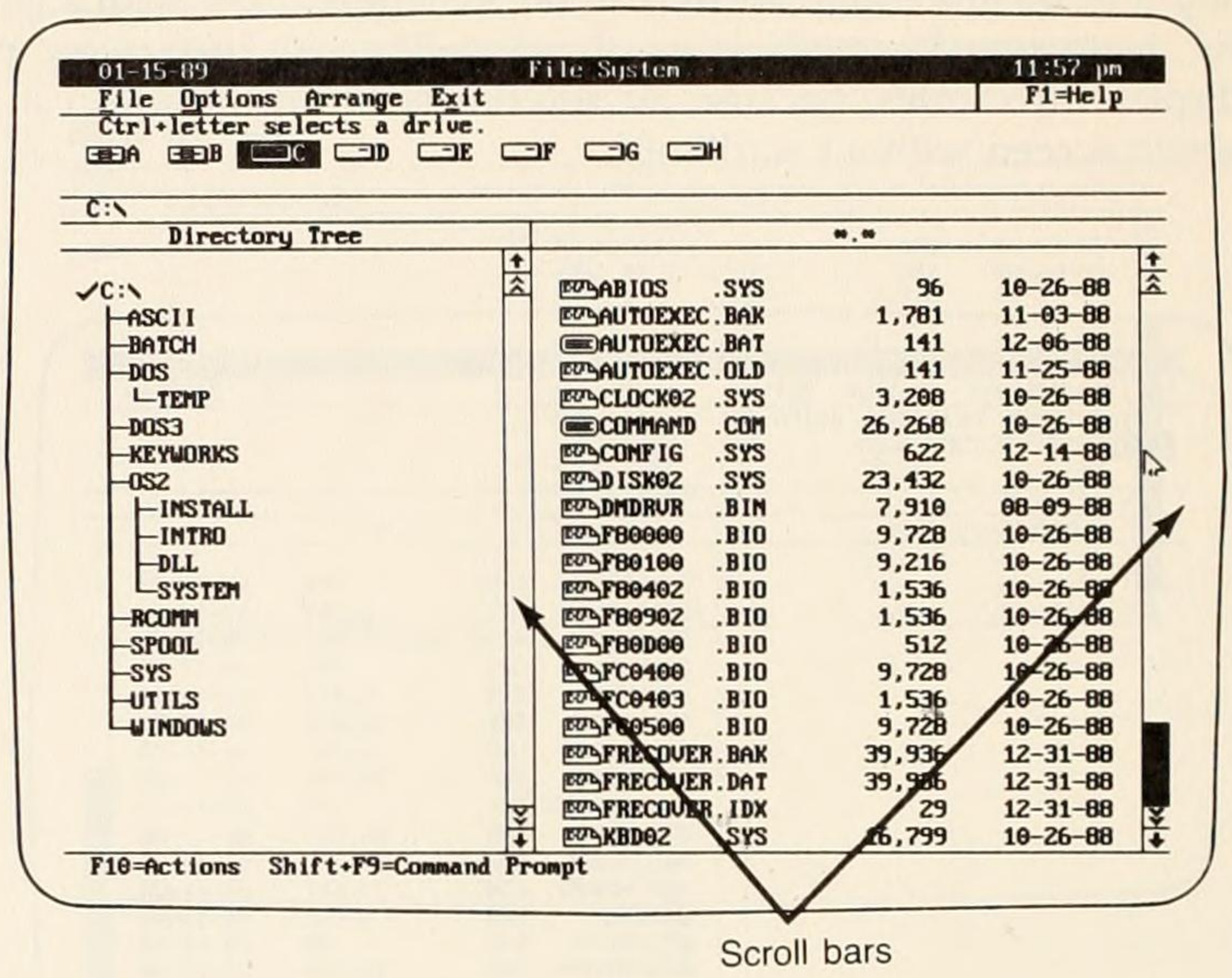


Fig. 16.5

Scroll bars are located on either side of the Shell screen.

The scroll bar has the following components:

- A single-line arrow box at the top and bottom of the bar. Clicking the cursor inside these boxes scrolls text up or down one line at a time.
- A page scroll box, indicated by double arrows, located below the top single-line box and above the bottom single-line box. Clicking these boxes moves text up or down one page at a time.
- A slider box located between the scroll boxes. You can position the cursor on this box, press and hold the mouse button, and drag the slider box up or down. The text window moves with the slider.

You use the scroll bar extensively with the DOS Shell File System. You can use the scroll bar in the next section as you explore the help screens. If you do not have a mouse, use the cursor keys and the PgUp and PgDn keys to scroll.

Using the Help System

Reminder:
The Shell help system is contextual; you get help on the currently highlighted menu item.

The Shell's help system is contextual. That is, DOS looks at the menu item currently highlighted and provides information about that selection. You can go from that help screen to other help screens to get help on additional topics. Figure 16.6 shows a typical help screen.

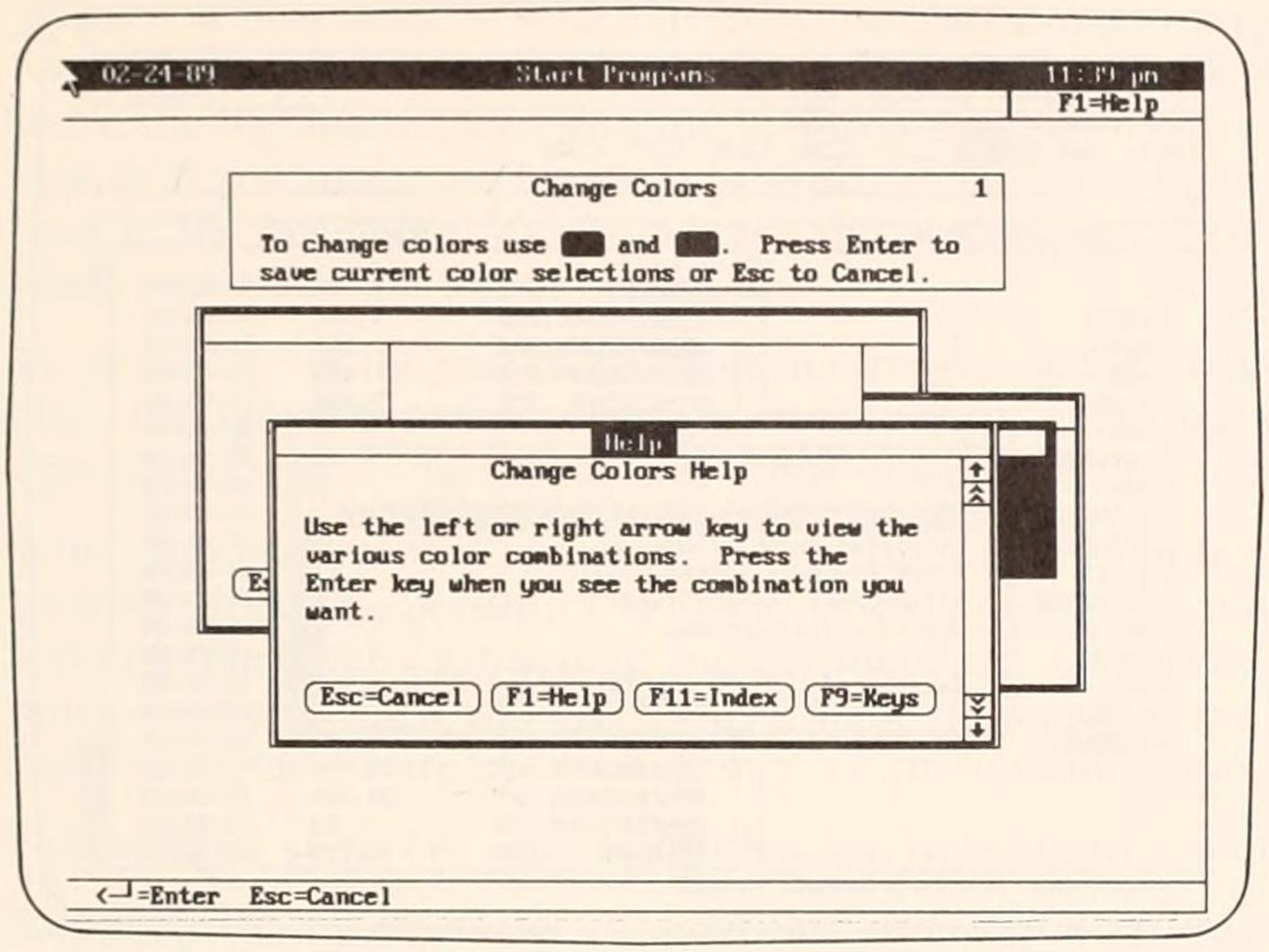


Fig. 16.6

The Change Colors help screen.

Notice that four boxes appear at the bottom of the help screen, labeled Esc = Cancel, F1 = Help, F11 = Index, and F9 = Keys. If you press F1 (or click the F1 = Help box), you receive help on how to use the help system. If you press F11 or click the F11 = Index box, you see an index of topics on which you can receive help. If you press F9 or click the F9 = Keys box, you see a list of special key assignments. Press Esc or click the Esc = Cancel box to return to the main menu.

Figure 16.7 shows the Index menu for the help system. You can use this screen to obtain an introduction to the help system or to get help on topics like the File System or Start Programs menu. Select the desired index item, read the help text, and when you finish with that selection, press F11 to return to the Index menu. If you press Esc, you return to the help screen from which you called the index.

The availability of help information varies depending on the menus you have on-screen. Note that when you add your own programs to the Shell, you can, if you want, type help instructions of your own. Then, anyone accessing that program can press F1 to see your special tips in the form of a help screen.

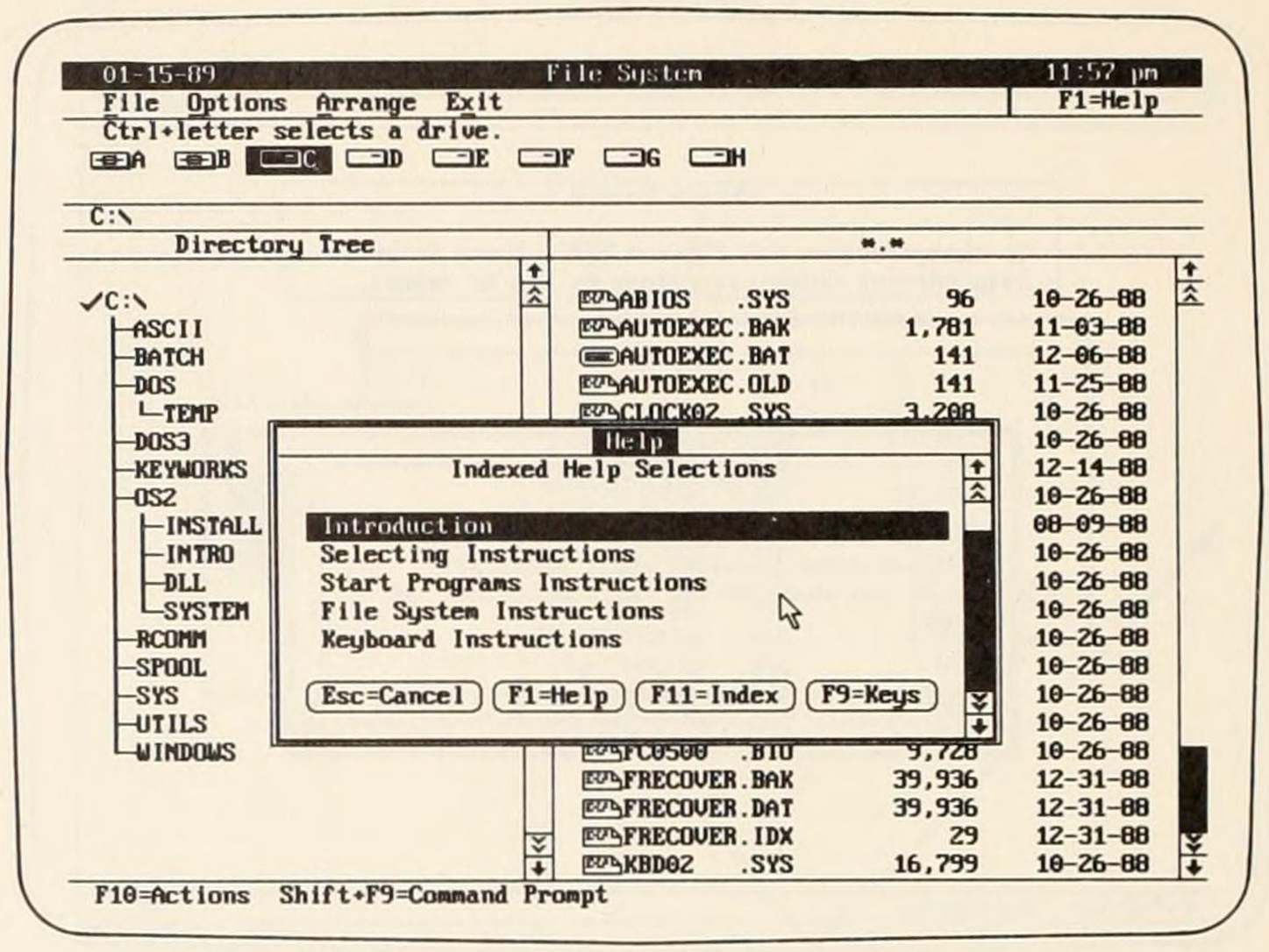


Fig. 16.7

The help system's Index menu.

Using the Start Programs Menu

Reminder:
Activate the Start Programs menu by pressing F10.

The Start Programs menu is the action bar at the top of the Shell screen. The menu can include a display of the date and time and a help prompt if you press F1. The action bar also shows three pull-down menu headings: Program, Group, and Exit. One of these is *highlighted*; that is, it displayed in a different color or in reversed tones. If none of the three items is highlighted, the Start Programs menu is not active. To activate the Start Programs menu, press F10. If a menu other than Start Programs is on-screen, select the Exit option from the action bar at the top of the screen to exit the current menu before pressing F10.

Reminder:
Make selections from menus by moving the cursor keys and pressing Enter, clicking the mouse button, or typing the underlined menu-option character.

Pull-down menus appear when you select a heading; they are lists of choices displayed beneath the headings. You can select options from the pull-down menus by using either the mouse or the cursor keys and Enter. To select a menu option with the cursor keys, press either the right or left cursor key to move the highlight from right to left. Press Enter when the choice you want is highlighted.

To select options with a mouse, move the cursor directly to one of the menu headings. The mouse cursor is a block in text mode and an arrow in graphics mode. Click the mouse button once to select the option you want.

Whether or not you have a mouse attached to your system, you can select a menu by typing the underlined letter associated with that menu heading, for example, Program, Group, or Exit.

When you have made a selection from the Start Programs menu or the action bar, the highlight drops down to the first item on the pull-down menu. You can move the highlighted line with the up-arrow and down-arrow keys or with the mouse. If you move the highlighting with the cursor keys, the highlight cycles back to the beginning when you reach the bottom of the list or to the bottom when you reach the top of the list.

To select one of the pull-down submenu items, highlight it and press Enter, or click the left mouse button. Alternatively, type the underlined letter of the option you want to select. To exit a pull-down menu, press Esc.

When you use the Shell, you may notice that some menu options are followed by three dots. The dots are DOS's way of telling you that this option is not an action in itself but a gateway to another menu. That submenu provides a list of functions from which you can choose. Menu options that are themselves headings for submenus are always indicated by three trailing dots.

Reminder: *Menu options followed by three dots display additional submenus.*

Follow these steps for a quick introduction to using the menus:

Step 1. *If the action bar is not activated (none of the three selections is highlighted), activate it by pressing F10.*

Step 2. *Use the left-arrow and right-arrow keys to move the highlight to the Program selection on the action bar. Press Enter, the down-arrow key, or P to select Program and activate the Program menu.*

Step 3. *When the pull-down Program menu appears, move the cursor to the Add... selection using the down-arrow key and presss Enter. Alternatively, press A.*

The Add Programs screen appears. Because you are just exploring menus and don't want to add a program at this time, press Esc to remove this display from the screen and return to the Main Group menu.

Step 4. *Pull down the Group menu. You can do this in one of two ways:*

- Press F10 to activate the action bar. Then move to the Group selection by using the right-arrow key and pressing enter, or simply by pressing G.

- If the Program menu is pulled down , move directly from the Program pull-down menu to the Group pull-down menu by pressing the right-arrow key. You don't have to go "through" the Group heading to reach the Group menu.

Step 5. *When the Group menu appears, move the cursor to the Add... selection and press Enter, or simply press A.*

The Add Group screen appears. Because you don't want to add any groups now, press Esc. The Main Group menu appears again.

Step 6. *Move the cursor to the Exit heading on the action bar. You can do this by choosing either of the options in step 4.*

The Exit menu appears with two choices: Exit Shell or Resume Start Programs. Move the cursor to Resume Start Programs and press Enter.

These steps have given you a quick tour through the basic Start Programs action bar menus.

Using Function Keys

The following sections describe the function keys you use with the DOS Shell program.

The F1 Key

If you need help, press the F1 function key. A series of text screens with useful information appears (refer to "Using the Help System" earlier in this chapter). You can page through the help screens with the PgUp and PgDn keys. When you are looking at help screens, you can use these keys:

Key	*Function*
Esc	Return to the menu
F1	Get instructions on how to use the help system
F9	Display a list of the special-key definitions
F11 or Alt-F1	Display an index of available help screens

The Shift-F9 Key

Reminder: *Use Shift-F9 to access DOS from the Shell.*

Any time you use the DOS Shell, you can perform a DOS function and then return to the DOS Shell. When you press Shift-F9, you see a DOS command prompt and a reminder that you can return to the Shell by typing **EXIT**.

When you access the DOS prompt in this way, the DOS Shell remains loaded and available for use. You can carry out DOS functions, or even load and run other programs if you have sufficient memory available.

Return to the Shell by typing **EXIT** at the DOS prompt and pressing Enter.

The F10 Key

The F10 key moves the cursor to the Start Programs action bar if the cursor is located in the Main Group menu, or to the Main Group menu if the cursor is located in the action bar.

Using the Program Submenu

To use the Program pull-down menu, select Program from the action bar. If the cursor is located in the Main Group menu, press F10 to activate the Start Programs action bar. Select the Program option from the action bar with the mouse or keyboard. The Program pull-down submenu looks like figure 16.8.

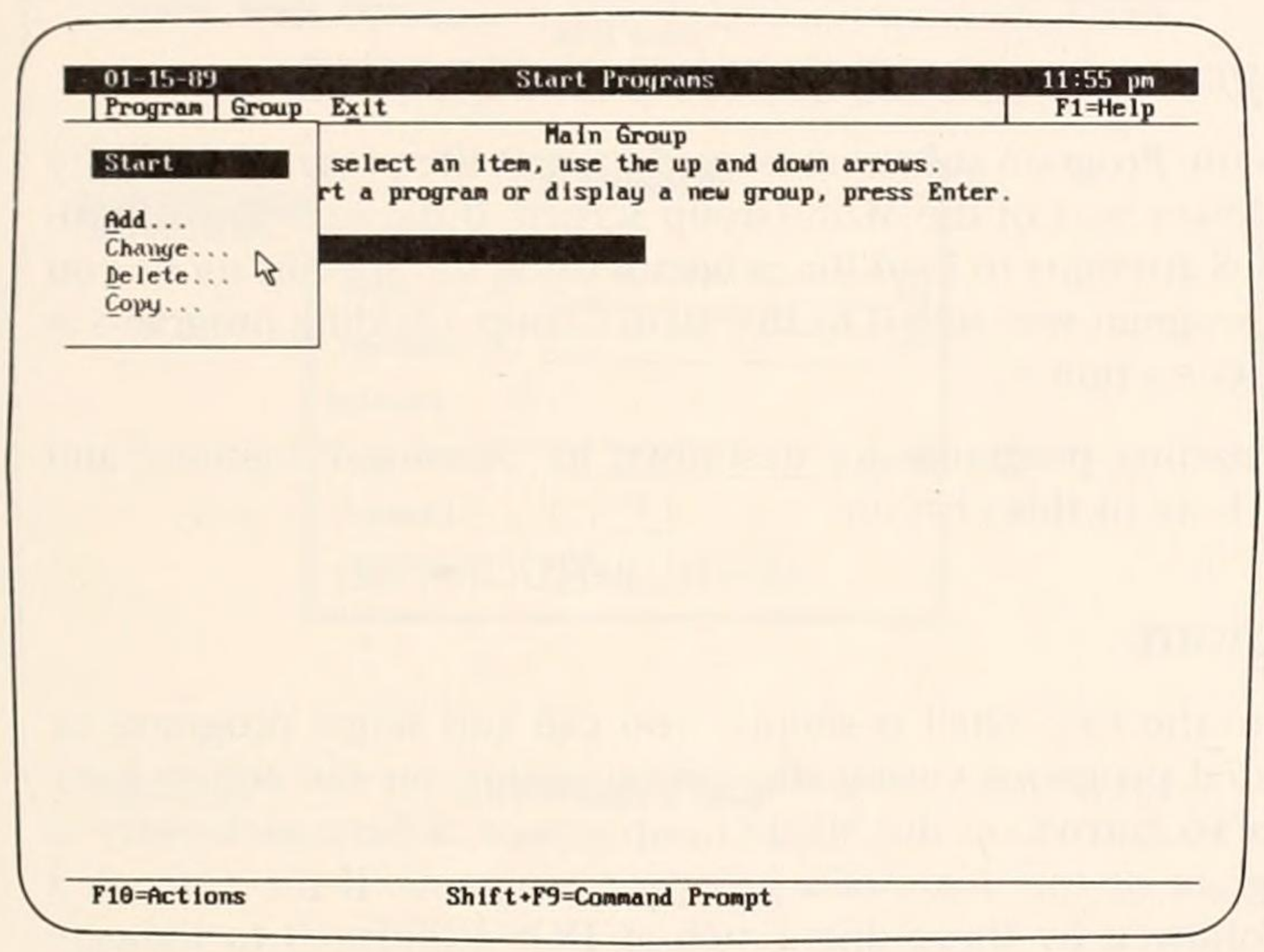


Fig. 16.8

The Program submenu.

If you use the Shell in graphics mode, some of the choices on the menu may be dimmed or blurred to indicate that they are not presently available. If you use the Shell in text mode, the underlined letter is replaced by an asterisk to indicate that the choice is not available or appropriate at the moment.

Reminder:
Some menu choices appear dimmed, or with underlined characters replaced by asterisks, to show they are not available.

Remember that you can choose a menu item by typing its underlined letter. You also can move the highlight with the cursor keys and press Enter to make a selection, or move the highlight with the mouse and click the mouse button.

The Main Group listing shows all the programs, or groups of programs, you can select from the Shell. Because this list is not a directory listing, you must add and delete the entries that appear on the list. You use the Program submenu to make these changes to the Main Group listing.

You use the Start option from the Program submenu to start the program highlighted on the Main Group screen. You use the Add... option to put additional programs in the Main Group list. You use the Change... option to modify the

guidelines or parameters used to load a specific program from the Shell. You can use this option to change certain information (such as the path DOS uses to find the program) about a program already available from the Shell. You use the Delete... option to remove programs from the menu entirely. You use the Copy... option to duplicate a given program in another menu. (You then can delete that program from the old menu if you want to move the program rather than simply duplicate it.) The following sections explore each of the choices on the Program submenu.

Starting a Program

The Start option on the Program submenu is used to start the program or utility highlighted on the lower part of the Main Group screen. If the item is an applications program, DOS attempts to load the program using the specifications you supplied when the program was added to the Main Group. (Adding programs is explained in the next section.)

Other methods of starting programs are described in "Starting Programs" and "Associating a File" later in this chapter.

Adding a Program

Cue: Use Add... to add a program or a group name to the Main Group screen.

Adding a program to the DOS Shell is simple. You can add single programs or you can group related programs (using the Group option on the action bar). You can have up to 16 entries on the Main Group screen, where each entry is either a single program or the name of a group of programs. If the entry is a group name, it is followed by three dots (such as DOS Utilities...) to indicate that a menu of choices is presented when that entry is selected. When you select the Add... menu option from the Program submenu, the screen shown in figure 16.9 appears.

Of the four parameters you can type on the Add Program panel, two are required and the other two are optional. The first required parameter is the title, or the label to appear on the Main Group screen. The title can have up to 40 characters and can contain spaces, uppercase or lowercase characters, or graphics characters.

The second required parameter is the set of commands needed to load the program. You can provide the commands in one of two ways: You can type all the commands needed to load the program, using up to 500 characters, on the `Commands` line. Alternatively, you can type all the commands in a batch file and call the batch file from the `Commands` line by typing **CALL** and the name of the batch file. Following is a sample batch file you could use to start the KEYWORKS program (located in the \KEYWORKS directory):

```
@ECHO OFF
C:
CD \KEYWORKS
KEYWORKS
CD \
```

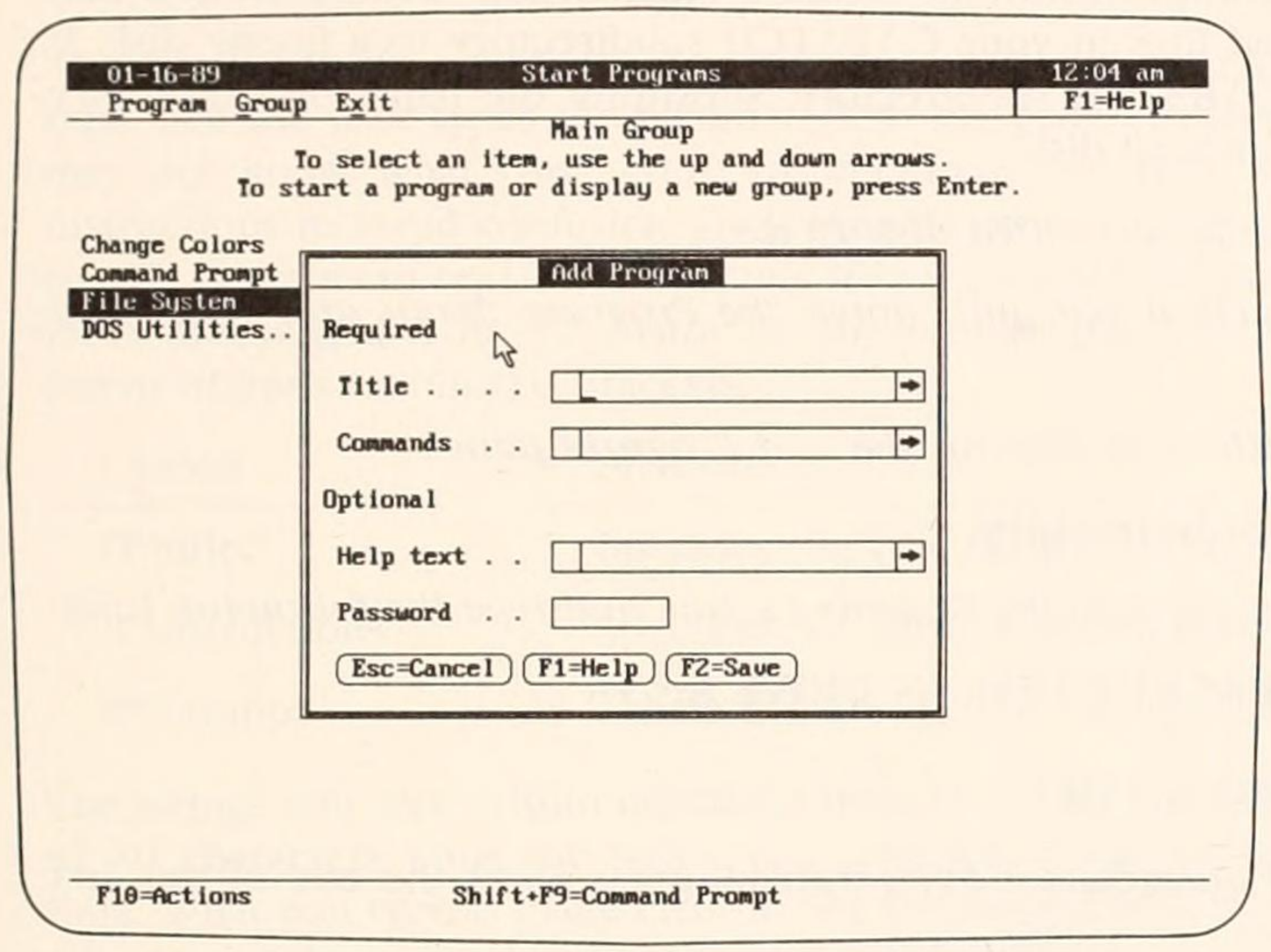


Fig. 16.9

The Add Program panel.

The Add Program panel enables you to provide two optional parameters. First, you can create your own help text, displayed when you highlight the title on the Main Group screen and press F1. You can type up to 478 characters of help text. The Shell neatly displays this information in pages you can scroll through. You may want to enter help text to explain what the program is or how it is to be used.

Cue:
Type your own help text when you add a program to the Main Group.

The second optional parameter you can provide is a password of up to eight characters. A password limits access to the program. If you provide a password, you must type it before the Shell will load, change, copy, or delete that program. Don't forget what the password is, or you cannot perform any of these functions!

The DOS Shell enables you to customize the Add Program entries even more extensively. Because you can create interactive menu selections (to ask for information "on the fly"), the Shell can be even more versatile than most batch files. The following sections explain how you can customize the entries you make with the Add Program panel.

Using Prompt Panels

Because you can tailor the parameters for each entry you add, you may create menu options for quite specialized operations. Using prompt panels, you can ask for information in a sophisticated manner.

The following steps explain how to create a Main Group menu selection that backs up all changed files in your C:\BATCH subdirectory to a floppy disk. If you don't have a C:\BATCH subdirectory, substitute the name of a directory you do have on your hard disk.

Step 1. *Insert a blank, formatted disk in drive A:.*

Step 2. *From the action bar, pull down the Program menu and select Add Program.*

Step 3. *Type the following title on the Add Program panel:*

Back Up Batch Subdirectory

Step 4. *Press Tab to move to the* `Commands` *line and type the following lines:*

ECHO INSERT BACKUP DISK IN DRIVE A:<F4>
PAUSE<F4>
XCOPY C:\BATCH A:\ /M

Do not type the characters *<F4>*; instead, press F4 at the end of the first and second lines.

Step 5. *Press Tab again to move to the* `Help text` *line and type the following:*

Copies any batch files that have changed since they were last copied. Files are copied to the floppy disk in drive A.

Step 6. *Press Tab one more time to move to the* `Password` *line and type this password:*

XCOPY

Step 7. *Click the F2=Save box or press F2 to save this new program.*

The Back Up Batch Subdirectory selection appears in the Main Group menu. Test this new menu item by selecting it a few times. To use this program, you must enter the password XCOPY. In "Changing a Program" later in this chapter, the Change option is used to remove the password.

You can create another menu entry that performs almost the same function as the Back Up Batch Subdirectory option created in the preceding steps. The new menu entry enables you to specify the source and destination drives and directories.

Cue:
Use square brackets in the `Commands` *line to create a prompt panel.*

First, you need to learn about the Shell's optional prompt panels. Prompt panels are created by typing a pair of square brackets ([]) on the `Commands` line of the Add Program panel. In the entry you just created, for example, you can prompt the user for the source and destination subdirectories by replacing *C:\BATCH* and *A:* with brackets:

```
XCOPY [] [] /M
```

The Shell displays an "Enter parameters" panel for each set of brackets you type, and the user types the parameters in the panel. Unfortunately, new users may not know what parameters are required; you need to provide specific instructions to avoid confusion. To solve this problem, the Shell enables you to specify prompts to replace the default "Enter parameters" panel with something more meaningful. You can customize the prompt panel by including three different options within the brackets:

Option	*Meaning*
/T"title"	Specifies the title for the prompt panel
/I"instructions"	Specifies any further instructions
/P"prompt"	Labels the line where the parameters are typed

The strings you type within quotation marks for the /T and /I options can be up to 40 characters long; the string you type for /P can be up to 20 characters long. What you type is made clear in the following example where you create a new Main Group option called *Back Up a Directory*:

Step 1. *Insert a blank, formatted disk in drive A:.*

Step 2. *From the action bar, pull down the Program menu and select Add Program.*

Step 3. *Type the following title:*

Back Up a Directory

Step 4. *Press Tab to move to the* `Commands` *line and type the following as one line, without pressing F4:*

XCOPY [/T"Source Directory" /I"Full drive/path of source directory" /P"Drive/path..."] [/T"Destination Directory" /I"Drive/path of destination directory" /P"Drive/path..."] /M

Step 5. *Press Tab again to move to the* `Help text` *line and type the following:*

Copies to the destination directory any files in the source directory that have changed since they were last copied.

Step 6. *Press Tab one more time to move to the* `Password` *line and type the following:*

XCOPY

Step 7. *Click the F2=Save box or press F2 to save this new entry.*

The Back Up Directory selection appears in the Main Group menu.

When you select this menu item, you are prompted with screens like those shown in figures 16.10, 16.11, and 16.12.

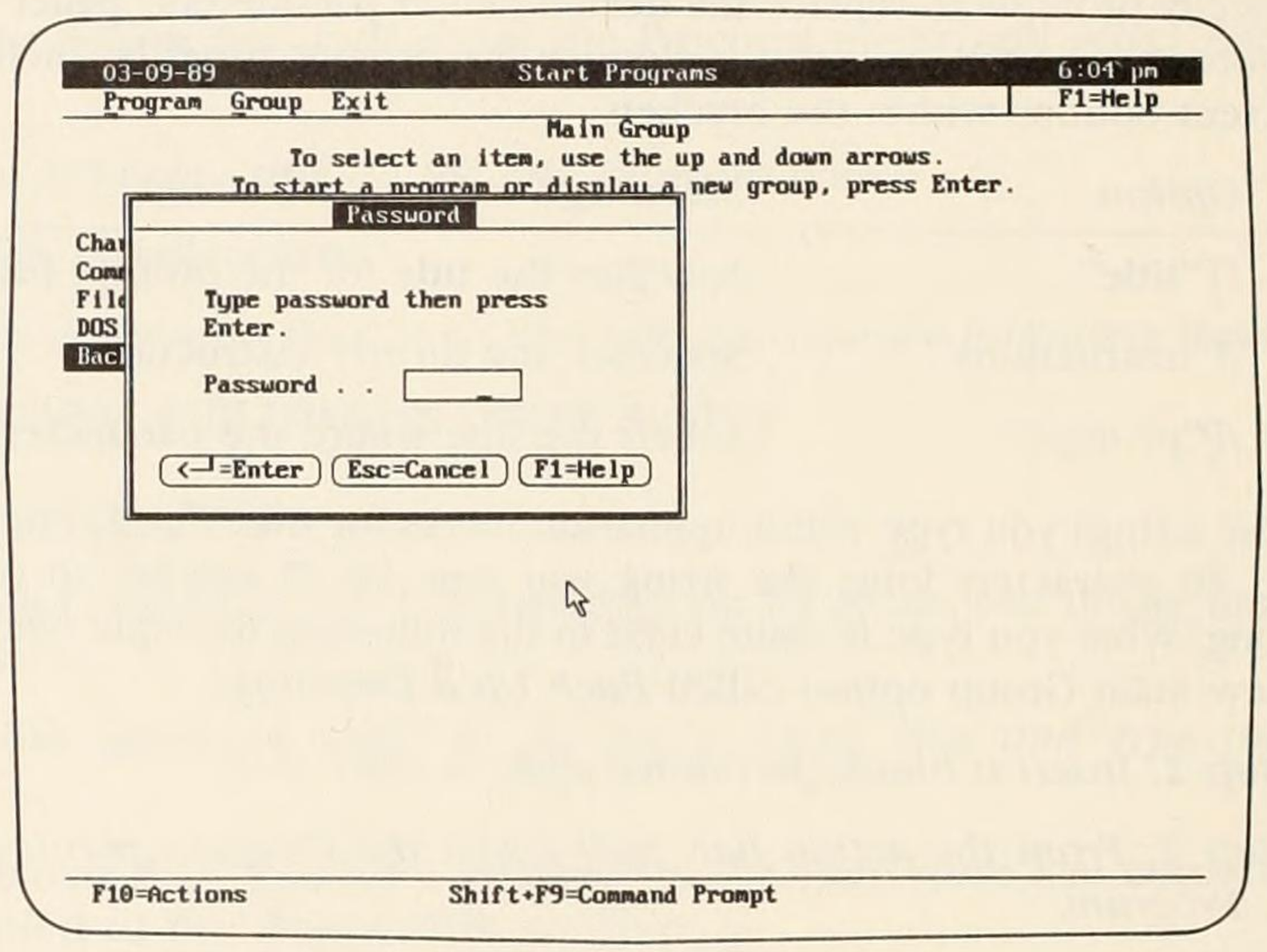


Fig. 16.10

The Password prompt panel.

In addition to the prompt panel, you can use a number of other commands to customize the Add Program parameters. These powerful facilities are described in the next section.

Using Additional Add Program Parameters

You can use additional parameters on the Add Program panel's `Commands` line to customize the program you add. This section describes the additional parameters you can use.

The /F"drive:\path\filename" parameter checks to see that the file specified by drive:\path\filename actually exists before you continue with the procedure. Suppose that you want to run a program from a floppy disk. Before continuing, you can check to see that the correct floppy disk is loaded.

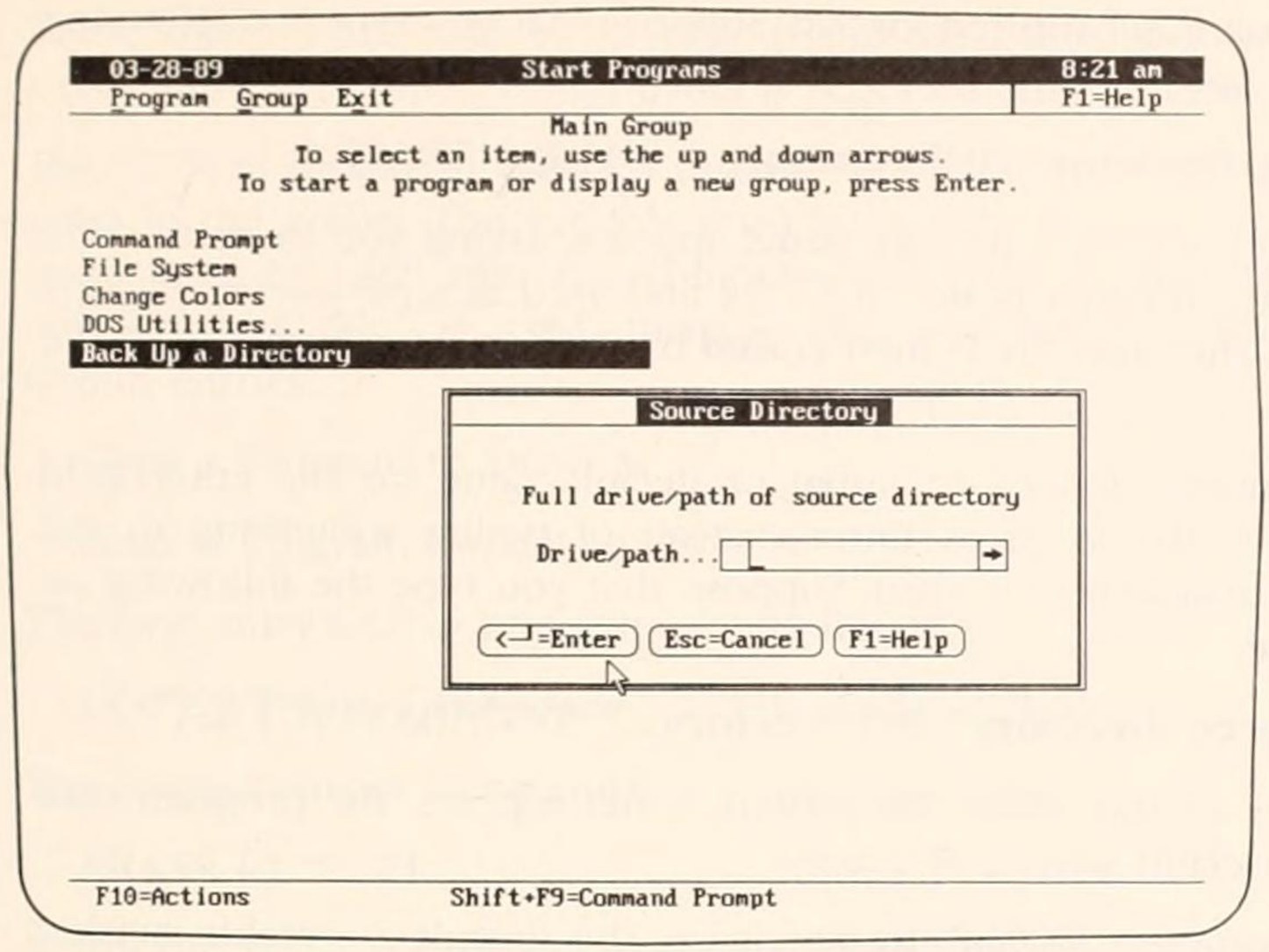


Fig. 16.11

The customized Source Directory prompt panel.

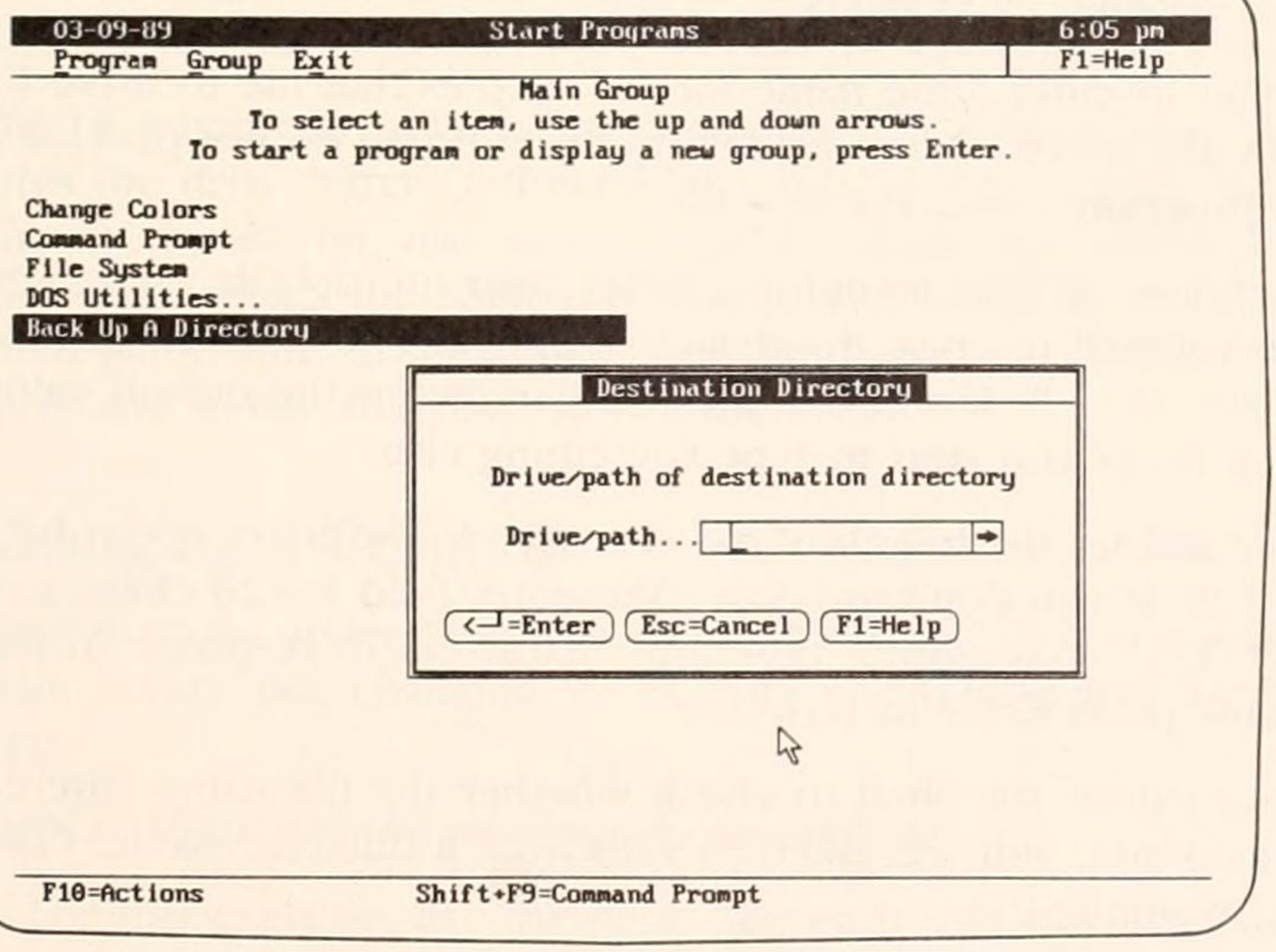


Fig. 16.12

The customized Destination Directory prompt panel.

The %n parameter is similar to the variable markers used in batch files. Replace the *n* with a whole number from 1 to 10 and place the parameter anywhere in the `Commands` line. When you enter a value for each parameter in the prompt

Cue:
Use the %n parameter in the same way you use variable markers in a batch file.

panels, the new value is substituted for %n. Suppose that you type the following on the `Commands` line:

COPY [/T"Enter filename" /P"Filename..." %1] B: ||ERASE %1

When this program is called, a prompt panel appears, asking you to enter a file name. The file name you type is stored in %1 and used as a parameter to copy the file to drive B:. The same file is then erased by the second command, ERASE %1.

The /D"text" parameter defines an initial or default value for the entry field within the brackets. If you press Enter instead of typing something in the prompt panel, the default text is used. Suppose that you type the following on the `Commands` line:

XCOPY [/T"Source directory" /P"Directory..." /D"C:\BATCH"] A:

If you do not enter a value when the prompt panel appears, the program uses C:\BATCH as the default source directory.

If you want, you can use /D"%n" to specify as the default a variable marker defined earlier in the command string. Suppose that the `Commands` line is as follows:

COPY [/T"Enter filename" /P"Filename..." %1] B: ||ERASE
[/T"Erasing File" /P"File..." /D"%1"]

This line prompts you to enter a file name for %1, copies that file to drive B:, and prompts you for the name of another file to erase. If you do not provide a new file name, the program erases the first file.

Cue:
Use /R to erase the default response as soon as you start typing a new response.

Sometimes, if you choose to type something other than the default value, you don't want to force yourself to erase the default supplied. The Shell provides a parameter to help you out: /R. If you include this parameter, the default value shown is erased as soon as you start to type something else.

The /L"n" parameter defines the length of the entry field. The letter *n* can be a number from 0 to 128. If you don't use /L"n", the entry field is 128 characters long. If you specify /L"0", you cannot type any characters in response to the prompt, but you must press Enter to proceed.

The /M"e" parameter causes the Shell to check whether the file name entered actually exists. If it doesn't, you are asked to substitute a different name. Consider the following example:

COPY [/T"Enter filename" /P"Filename..." /M"e" %1] B:

NOTE: The /M switch used with the XCOPY command in the preceding section is different than this /M"e" parameter. The earlier XCOPY switch appeared outside the brackets and did not specify "e".

These instructions take effect only if the file name specified by %1 actually exists. Otherwise, the prompt panel is repeated until you enter a valid filename.

The /C"%n" parameter passes a variable defined in one program to another program in the group. The variable must already be defined; that is, the first program must be used after the computer is turned on and before the program containing /C"%n" is used. Imagine that you have the following two Main Group entries:

Copy a Program to Drive A:

Erase a Program Copied to Drive A:

The first entry's `Commands` line looks like this:

COPY [/T"Enter filename" /P"Filename..." %1] A:

The second entry's `Commands` line looks like this:

ERASE [/C"%1"]

If you run the first program, you don't have to enter the name of the file to be erased; the name of the file you copied is supplied as the default. The %1 parameter retains the value of the %1 defined by the most recently used program. If you had another menu option that used %1, and that option was selected more recently, its value for %1 would be used in the ERASE option. If you use /C"%n", you may want to pair the options that use it.

The /# parameter must be used outside the prompt-panel brackets. /# substitutes the drive letter (followed by a colon) from which the Shell was started. The /@ parameter also must be used outside the brackets; it substitutes the path from which the Shell was started. No backslash is included in the substituted path name. You can use both the /# and /@ parameters to default to the drive and directories used to run the Shell.

Reminder:
Use the /# and /@ parameters outside the square brackets.

Changing a Program

Now that you understand how to use the Add... option to add a program to the Main Group list, changing an existing program should be easy. Follow these steps:

Step 1. *Highlight the program to be changed.*

For this exercise, use the Back Up Batch Subdirectory program you created earlier. You cannot change any of the default menu entries, such as Command Prompt, File System..., DOS Utilities..., or Change Colors; you can change only those entries you have added yourself.

Cue:
You can change only those menu entries you have added; you cannot change the default Shell entries.

Step 2. *Pull down the Program menu and select Change....*

The Change Program panel appears. This panel looks like the Add Program panel shown in figure 16.9, except for the title. Existing information about the selected entry appears on the various lines of this panel.

Step 3. *Change any aspect of the entry you want.*

In this example, press Tab to move the cursor to the `Password` line and delete the password.

Step 4. *Click the F2=Save box or press F2 to save the changed entry.*

Now you can use the Back Up Batch Subdirectory entry without entering a password.

Deleting a Program

Cue: *When you delete an entry, you delete only the name from the list; you do not delete the file.*

You may want to delete from the Main Group list a program that you no longer use. The program itself is not removed, only the menu entry. You cannot delete the default menu entries such as Command Prompt, DOS Utilities..., or Change Colors. To delete other programs, follow these steps:

Step 1. *Highlight the entry you want to delete.*

For this exercise, select the Back Up Batch Subdirectory program you created earlier.

Step 2. *Pull down the Program menu and select Delete....*

The Delete Program panel appears. This panel asks you to confirm or abort the process. Delete Program is the default option.

Step 3. *Press Enter to delete the selected entry.*

The Back Up Batch Subdirectory has been removed from the Main Group list.

Copying a Program

Cue: *You can copy an entry to the same group or to another group.*

You can copy a Main Group entry if you want; the DOS Shell has no prohibition against having two programs in the same group with the same name, but such an arrangement has little point. You can also copy a program to a subgroup. In this exercise, you copy the Back Up a Directory entry you created earlier to another the DOS Utilities group:

Step 1. *Highlight the entry you want to copy in the Main Group list.*

For this example, highlight Back Up a Directory.

Step 2. *Pull down the Program menu from the action bar and choose Copy....*

Step 3. *Highlight the group where you want to copy the entry.*

In this example, move the cursor to DOS Utilities... and press Enter; alternatively, double click with the mouse. The DOS Utilities menu appears.

Step 4. *Click F2=Copy or press F2 to copy the program to the new group.*

Back Up a Directory now appears both in the Main Group and the DOS Utilities menus.

Using the Group Submenu

The Group submenu is easy to use once you know how to add, change, delete, and copy programs from the Main Group list. As you will see, groups in the Main Group can have subgroups of their own. You may find yourself building menus several levels deep so that you can group a large number of programs together conveniently. The Group submenu is shown in figure 16.13.

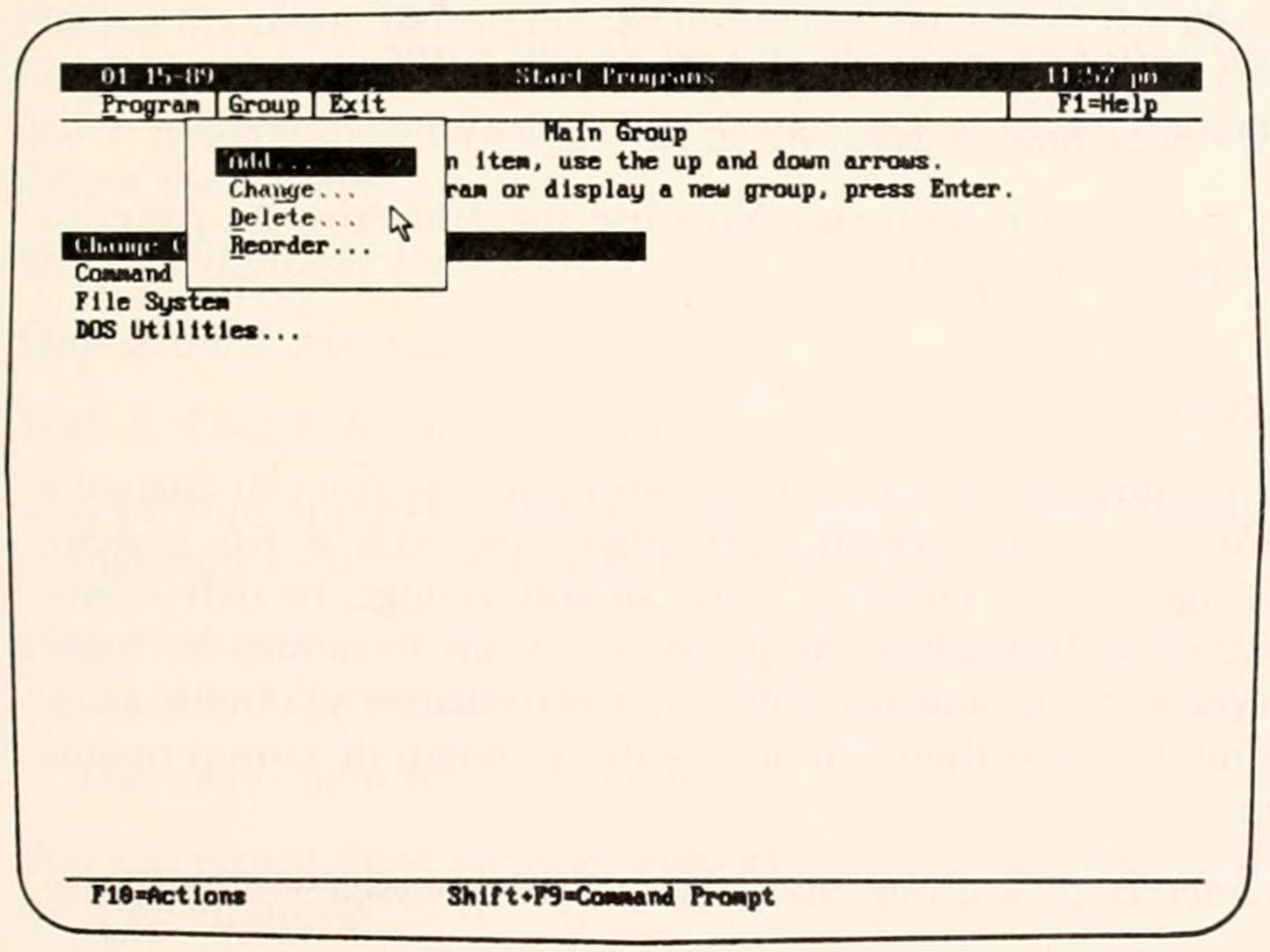


Fig. 16.13

The Group submenu.

Notice that the Group submenu has four choices. Three of them will be familiar: Add, Change, and Delete. You learned to use these options on the Program submenu. The fourth, Reorder, is a new option that enables you to change the arrangement of programs in the group listing. Each option is described in the following sections.

Adding a Group

Cue:
You can add a group if fewer than 16 entries appear on the menu.

You can add a group to a menu if fewer than 16 entries appear on the Group menu with which you are working. To add a group, follow these instructions:

Step 1. *Pull down the Group menu from the action bar.*

Step 2. *Choose the Add... option.*

The Add Group panel appears. This panel contains these four entries: `Title`, `Filename`, `Help text`, and `Password`.

Step 3. *Enter a title.*

The title is the name that appears in the Group menu and can be up to 37 characters long. The Shell adds the trailing dots automatically.

Step 4. *Type a file name.*

The file name is the name under which this new menu is to be stored. The file name can have eight characters; the Shell automatically adds the .MEU extension for you.

Step 5. *Add optional help text and a password, if you want.*

Step 6. *Click the F2=Save box or press F2 to save the new group entry.*

The new title appears in the Group menu. Now use the Add Program panel to add program files to this group.

Changing a Group

Cue:
You can change the parameters of a group you have defined.

You can change the parameters used to define a group just as you can change a program. You cannot change the default DOS subgroups, such as File System, DOS Utilities, or Change Colors; however, you can add, change, or delete programs within these groups. Just select the group you want to modify by highlighting it in the current group and pressing Enter. Alternatively, double click the group with the mouse. You then can access the Program or Group menus within the subgroup.

To change the parameters of a group you have added yourself, follow these directions:

Step 1. *Highlight the name of the group.*

Step 2. *Pull down the Group menu from the action bar.*

Step 3. *Select the Change... option.*

The Change Group panel appears. This panel looks like the Add Group panel, except that the current information about the selected group appears in the fields.

Step 4. *Use the Tab key to move to the field you want to change and make your changes.*

Step 5. *Click the F2=Save box or press F2 to save the changes.*

Deleting a Group

You can remove a group you have added yourself with the Delete... option in the Group menu. Follow these directions:

Step 1. *Highlight the name of the group to be deleted.*

Step 2. *Pull down the Group menu.*

Step 3. *Select the Delete... option.*

Step 4. *Confirm that you want to delete the group by pressing Enter.*

The group is removed from the menu.

Changing the Order of a Group's Listings

Cue: *Rearrange the order of the items in a menu to suit your needs.*

Placing the menu items you use most near the top of the group listing is convenient; you can find these items more quickly. You can place menu items in any order you want. To move an entry from one place to another in the menu list, follow these steps:

Step 1. *Highlight the menu item to be moved.*

Step 2. *Pull down the Group menu.*

Step 3. *Choose the Reorder... option.*

A prompt appears on-screen under the group title, telling you to move the highlight to the new position and press Enter.

Step 4. *Use the cursor keys or the mouse to move the highlight to the position you want the selected menu item to appear in the group. Press Enter.*

The menu item has been moved.

You can repeat these steps as often as necessary to produce the order you want for a given group.

Creating a New Main Group Menu

You may want to create an entirely new Main Group menu. Perhaps you want to move the Command Prompt option to a special subgroup that has password protection. If you make this change, users who know the password can use the command prompt, and others cannot.

You may find that certain Main Group entries, such as Change Colors, are rarely used. The DOS Shell does not allow you to delete entries from the Main Group, however. What do you do? The following exercise shows you a way around that roadblock.

In this exercise, you create a new subgroup that includes all the programs you want in your Main Group except for the DOS defaults. Then you cause the Shell to use the new subgroup as the Main Group instead. You haven't deleted anything from the Main Group; you have just scrapped it and substituted a new one with the files you want. The old Main Group can become a subgroup under the new Main Group. As confusing as this may sound, the procedure is relatively simple, as you see in the exercise that follows:

Cue:
The first step in creating a new Main Group menu is to copy all the programs you want to the new group.

Step 1. *With the Main Group menu on-screen, pull down the Group menu from the action bar.*

Step 2. *Choose the Add... option.*

Step 3. *Type a name for the new group that is to become the new Main Group.*

For this example, type **NEW** as the name. The Shell later substitutes the name *Main Group* for you, so the name you choose at this point is almost irrelevant.

Step 4. *Press Tab to move to the* `Filename` *line. Type* **NEW** *as the name under which to save this menu. DOS adds the extension .MEU for you automatically.*

Step 5. *Type any help text you want. Do not type a password; putting a password on the menu that will be your Main Group is not a good idea.*

Step 6. *Click the F2=Save box or press F2 to save this entry.*

Your new group appears in the present Main Group listing. Next you copy the programs you want to appear in the new Main Group (called NEW) from the existing Main Group.

Step 7. *Highlight a program you want to copy.*

The program can be in the current Main Group, or it can be in one of the other subgroups you have created. The program cannot be one of the default DOS programs, such as Change Colors.

Step 8. *With the group where the program to be copied is located on-screen, pull down the Program menu from the action bar.*

Step 9. *Choose the Copy... option.*

A prompt reminds you to press F2 when the group to which you want to copy the program is on-screen.

Step 10. *Move to the NEW group.*

If you are copying a program from a subgroup, press Esc to exit from the subgroup and return to the current Main Group. Then select the NEW group with the cursor or mouse. If you are copying a program from the Main Group, simply choose the NEW group.

Step 11. *Press F2 when the NEW group is on-screen.*

The selected program is copied to the NEW group.

Step 12. *Repeat steps 7 through 11 until all the programs you want copied to the NEW group have been duplicated.*

Now you must substitute the NEW group for the current Main Group. The Main Group is stored in a file called SHELL.MEU. Your DOSSHELL.BAT file contains the following line:

Cue:
The second step in creating a new Main Group menu is to alter the DOSSHELL.BAT file.

```
/MEU:SHELL.MEU
```

This statement tells the Shell to load the SHELL.MEU file as the Main Group when the Shell starts up. To use the NEW.MEU file instead, edit DOSSHELL.BAT and substitute *NEW.MEU* for *SHELL.MEU* in this statement. You can use EDLIN or any other ASCII text editor to edit batch files. To do this, follow these steps:

Step 1. *Exit from the Shell (use the Exit option on the Main Group).*

Step 2. *Load DOSSHELL.BAT into your text editor. Find the line containing /MEU:SHELL.MEU. Replace SHELL.MEU with NEW.MEU and save the file.*

Step 3. *Reload the DOS Shell using DOSSHELL.BAT.*

The NEW group is now the Main Group menu and is labeled as such. However, you don't have access to any of the default DOS functions formerly available from the Main Group. To resolve this problem, add the old Main Group as a subgroup to the new Main Group with these directions:

Cue:
The third step in creating a new Main Group menu is to add the default Shell utilities to the new group.

Step 1. *From the new Main Group, pull down the Group menu.*

Step 2. *Select the Add... option.*

Step 3. *Type a name for the old Main Group. For example, type* **Shell Utilities**.

Step 4. *On the* `Filename` *line, type* **SHELL.MEU**.

Step 5. *Type any help text you want.*

Step 6. *If you want to protect this group with a password, type one on the* `Password` *line.*

Step 7. *Click the F2=Save box or press F2 to save the new group parameters.*

The Shell Utilities group appears in the new Main Group. Shell Utilities contains the old DOS Shell functions like Change Colors and DOS Utilities.

Using the Exit Submenu

The third pull-down menu in the Start Programs action bar is the Exit menu. Two options are provided on this menu: you can either exit the DOS Shell entirely, or you can return to the Start Programs action bar.

If you elect to leave the DOS Shell, you cannot return to it unless you reload it by typing the DOSSHELL command on a DOS prompt. If you want to leave the Shell temporarily, use the Shift-F9 key described earlier in this chapter.

Using the Main Group Menu

The following sections explain the various options on the Main Group menu.

Starting Programs

You don't have to start a program from the Program submenu. In fact, you may seldom start a program from the action bar; you can use a much faster method.

Cue: Using the Main Group menu to start a program is faster than using the action bar.

You can start programs directly when the Main Group menu is active. Simply highlight the selection you want and press Enter, or click the mouse button. The program starts exactly as if you had chosen it using the Start option from the Program submenu.

Using the Command Prompt

The Command Prompt option in the Main Group menu enables you to access the DOS V4 prompt temporarily. After you finish executing DOS commands, return to the Shell by typing **EXIT** at the DOS prompt and pressing Enter.

Using the File System Menu

The DOS Shell File System is available from the Main Group menu and enables you to perform a long list of functions with your files. Accessing a directory of those files is the most rudimentary of those functions. The directory feature is a good introduction to the File System.

To access the directory, select File System from the Main Group menu. DOS reads the current disk and displays the screen shown in figure 16.14.

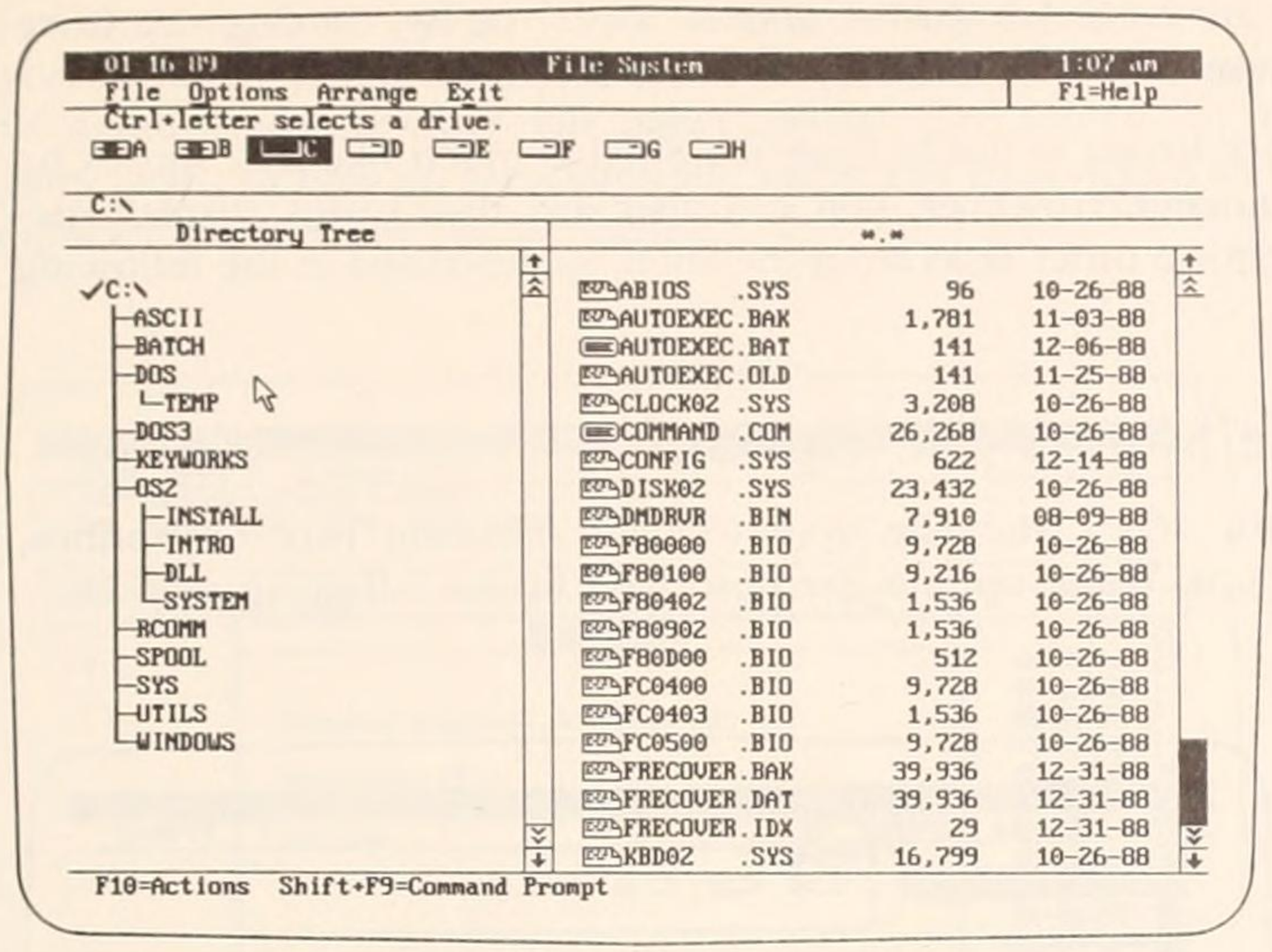


Fig. 16.14

The File System menu.

The menu displayed has four options: File, Options, Arrange, and Exit (each of these options is described in the following sections). The display shows letters or icons (depending on whether your Shell is in graphics or text mode) representing the disk drives available in your system. The current disk drive is highlighted.

The split-screen display shows you two key pieces of information. On the left is a directory tree that shows how the files are arranged on your disk.

Cue:
The File System screen is split to show a directory tree and a list of files in the current disk drive.

The right side of the display lists the files on the current disk drive. This list looks much like that shown by the DIR command from the DOS prompt. However, each file has an icon or character next to it that indicates the type of file it is (if you use graphics mode). The icons look like miniature documents. This side of the display gives the same information available from the DOS prompt: the file name, extension, size, date, and so forth.

In many cases, not all the files on a disk can be displayed on-screen at once. The DOS Shell provides several ways to scroll back and forth through a listing of file names. If you have a graphics-based system and a mouse, you can use the scroll bar. If you do not have a mouse or if you are using a text-based Shell, use the cursor keys. Before using either method, you must make sure that the file-display panel is active by highlighting an option on the panel.

To view a directory of a different disk, select from the third line of the display the letter or icon representing the desired disk drive. If the drive area is active, you can select a drive by holding the Ctrl key and pressing the letter corre-

sponding to that drive (Ctrl-A, Ctrl-B, and so forth) or by clicking the drive icon with the mouse. Press the Tab key to make the drive area active.

So far, the directory listing available from the Shell is much like that shown by the DOS DIR command. However, you can also sort directories, display specified files, and perform other tasks from the Shell, as described in the following sections.

Using the File System File Option

As shown in figure 16.15, the File System's File submenu lists 12 options, divided into 3 groups. These options are described in the following sections.

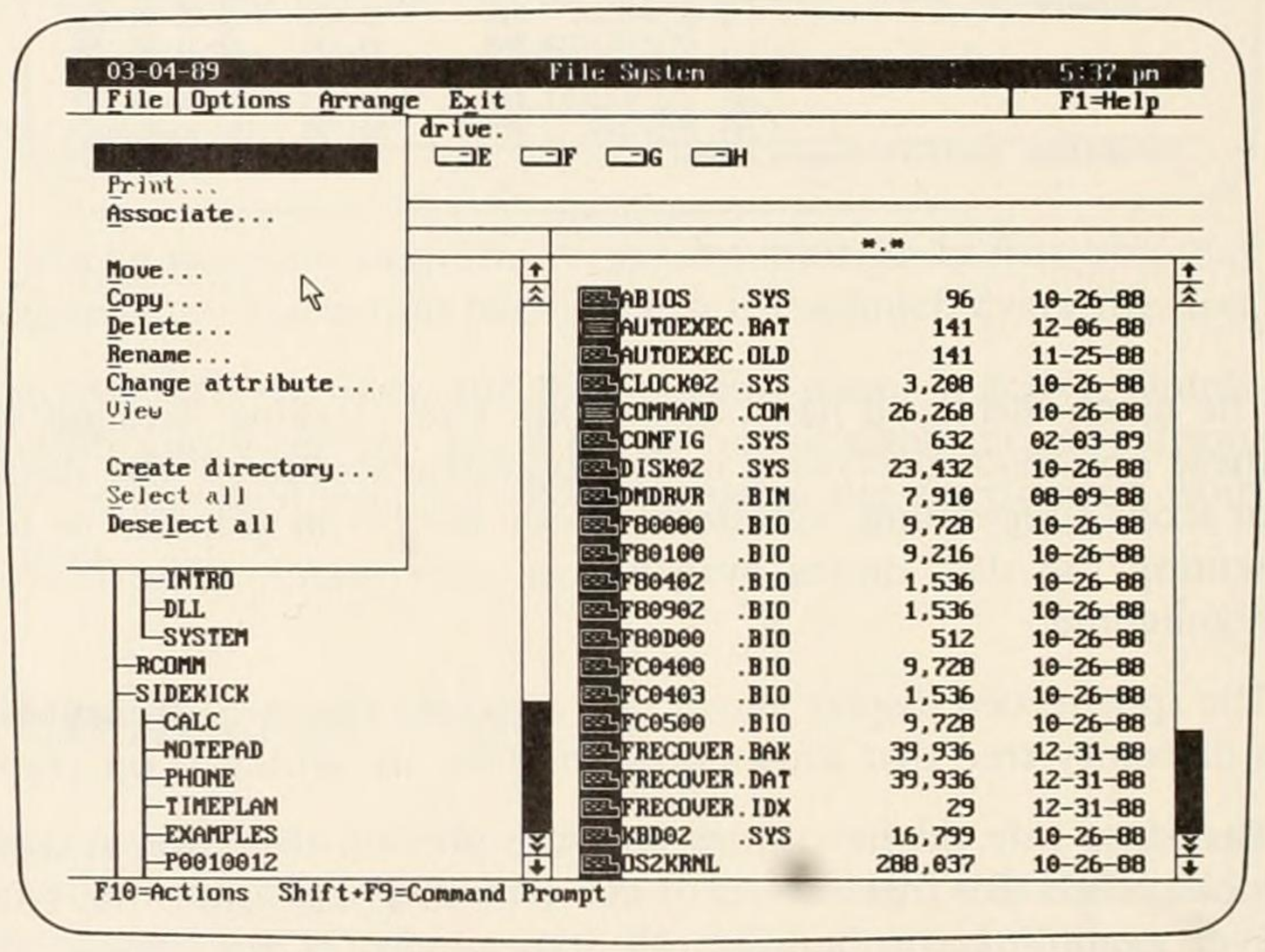


Fig. 16.15

The File submenu.

Opening a File

Cue:
You can highlight and start a program file if it has the extension .EXE, .COM, or .BAT.

You can start a program by double clicking the mouse pointer on an appropriate file entry in the File System directory listing. If you don't have a mouse, use the arrow keys to highlight the desired entry and press Enter. DOS runs the file if it is a valid .EXE, .COM, or .BAT file. If you have used the Associate... option (described later in this chapter) to link a file extension to an application, you can start a program by clicking a data file.

You also can start a program by choosing the Open... option from the File menu when a file is highlighted in the directory listing. Alternatively, you can type a file name when the Open File panel appears (see fig. 16.16). In the Open File panel, you can type an associated file name and the options to be passed to the application when the file is opened.

Cue:
You can start a program by highlighting the file name and choosing Open... or by choosing Open... and typing the file name.

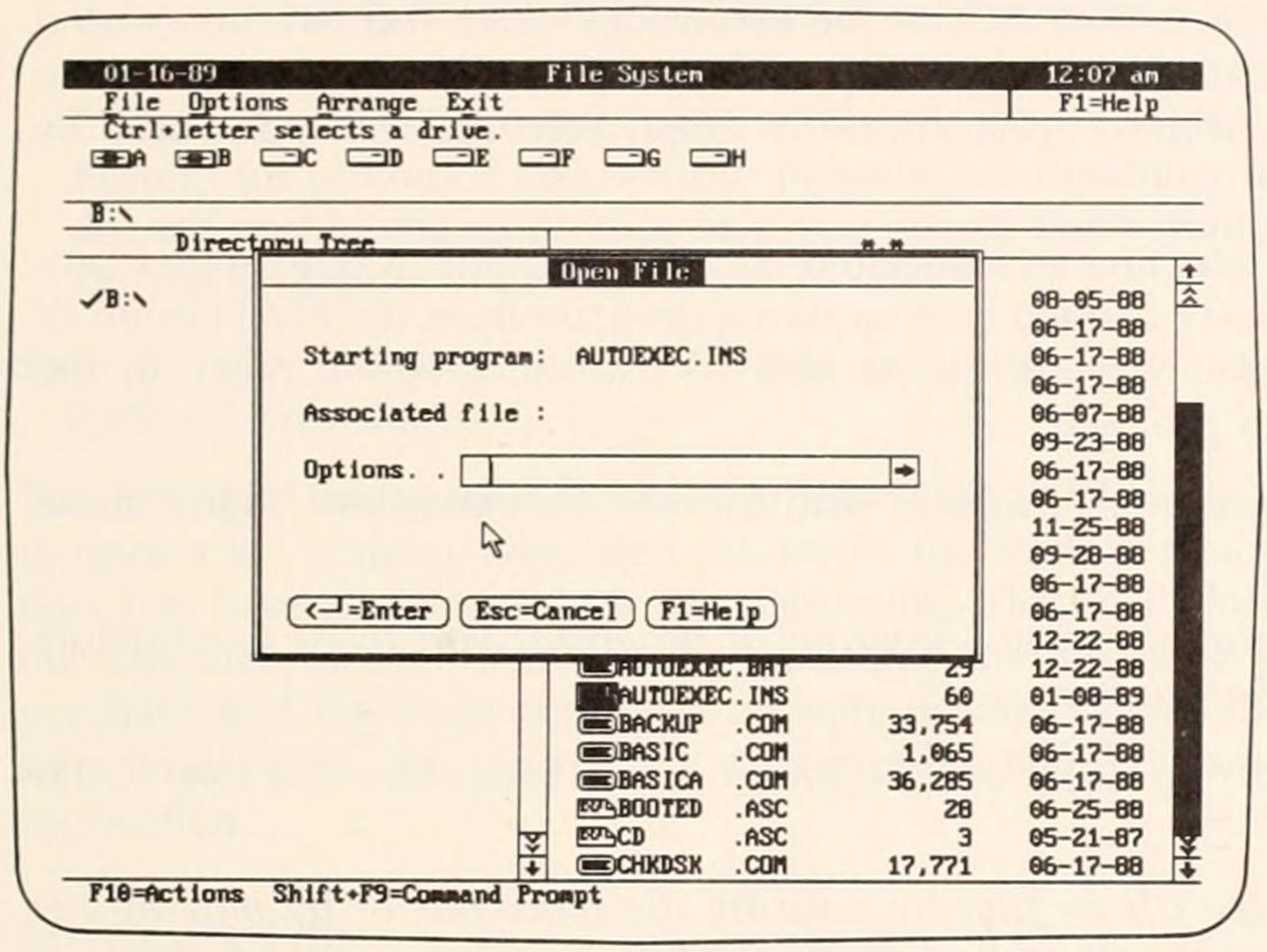


Fig. 16.16

The Open File panel.

Printing a File

The DOS Shell File System includes a Print... option, similar to the one available at the DOS prompt except that the switches used at the prompt are not readily available in the Shell. You could create a separate program that includes the PRINT command along with parameters that enable you to enter switches. The Shell's Print... option, however, is generally used only to get a quick printout of an ASCII file, so creating such a program is probably not worth your while.

Cue:
Use the File System Print... option to obtain quick printouts of ASCII files.

In ASCII mode, the File System's Print... option works quite well. You must load PRINT.COM from the DOS prompt for the option to work, however (Chapter 9 describes the PRINT command). Once you have loaded PRINT.COM, follow these steps to print a file:

Step 1. *Mark in the File System's directory listing the file or files you want to print.*

Step 2. *Pull down the File menu.*

Step 3. *Choose Print....*

Each file you mark is submitted to DOS for printing. A panel confirming that the file has been submitted for printing appears, displaying the name of each file in turn.

Associating a File

Cue: *Associate a particular file extension with an application program to simplify the starting of the program.*

The Associate... option enables you to specify to the Shell which applications programs are associated with various file extensions. After you have associated particular applications programs with particular file extensions, you can automatically start one of those applications by simply selecting a data file used by the application.

This procedure of selecting an associated data file is a third way to start a program. You have already learned how to start a program from the Main Group or subgroup listings, and you also know how to use the Program menu in the action bar to start a program.

To associate an applications program with specific file extensions, follow these steps:

Step 1. *Choose the applications program or programs you want to associate with specific extensions.*

The programs can be .COM or .EXE files, or .BAT batch files that load .COM or .EXE files.

Step 2. *Highlight the file or files by pressing the space bar or by moving the mouse pointer to the file's name and clicking.*

Cue: *You can associate up to 20 extensions with your applications programs, but no two applications can be associated with the same extension.*

You may have to change the current disk drive or directory to locate the application. Ideally, the applications programs you select use a single default file extension, such as .PCX for Publisher's Paintbrush or PC Paintbrush, .DOC for DisplayWrite 4, .WP for WordPerfect, .WKQ for Quattro, .BAS for BASIC programs, and so on. You can associate a total of 20 extensions with your applications programs where no two applications are associated with the same extension.

Step 3. *Pull down the File menu in the action bar and choose the Associate... option.*

If you have marked several files, the first file name appears. If the file does not have a valid extension (.COM, .EXE, or .BAT), you are told that the extension is invalid and are asked to choose between skipping the file or entering a different extension.

Step 4. *If the file is a valid application or batch file, use the* `Extensions` *line to type the extension or extensions you want to associate with that application.*

Type only the one to three characters of the extension, not the period. If you type more than one extension, separate the entries with a space.

Step 5. *When you have typed the extensions for a given application, press Enter.*

The Prompt for Options panel appears. In this panel, you specify whether you want the Shell to pause and ask for options when it loads the application or load the application directly. Some programs enable you to specify something at the DOS prompt, such as the file to be edited. You can choose Prompt for options if you want to provide this capability. If your application doesn't enable you to do this, or if you simply don't want this option, select Do not prompt for options.

Step 6. *After choosing either Prompt for options or Do not prompt for options, press Enter.*

The Associate information is stored in a file called SHELL.ASC. Any time you try to open a file with an associated extension, the Shell instead opens the application you have associated with that extension. The Shell doesn't actually open the data file you have selected. You can type the file name when prompted, if you have told the Shell to prompt for options when a data file with that extension is selected. The DOS Shell cannot pass the file name directly to the application.

Moving a File

You can select the Move... option from the File menu if you need to move a file or files quickly from one subdirectory to another or if you need to send a file to a disk for archiving.

The Move... option works just like the Copy... option (described in the next section), except that after you have moved the file to the new directory or disk, the file is erased from the old location. The names of the files you have marked in the file listing appear on the `From:` line; specify the destination on the `To:` line.

Cue:
Use the Move... option if you want to copy a file to a new disk or directory and then erase the original file.

If you have activated the Confirm on Delete choice in the File System Options menu, the Shell asks you for confirmation before removing the old file.

Copying a File

You already should be familiar with the DOS COPY command (described in Chapter 7). Use these instructions if your system has two floppy disk drives. You should have the DOS Shell loaded and the Start Programs menu on your screen.

Step 1. *Place into drive A: a blank, formatted disk.*

Step 2. *Place into drive B: a disk containing files you want to copy.*

Step 3. *Select File System from the Main Group menu.*

The File System menu appears. Make sure that drive B: is selected. If it isn't, select it with the mouse, or hold the Ctrl key and press the letter B.

Step 4. *Select the File menu from the action bar.*

Press F10, if necessary, to activate the action bar. You can pull down the File menu with the mouse, or you can type the letter F to select the menu. Note that the Copy... option is blurred, shaded, or marked with an asterisk, depending on your display mode.

Step 5. *Choose the Select all option from the File menu.*

The Copy... option becomes available.

Step 6. *Choose the Copy... option.*

You see a panel like the one shown in figure 16.17.

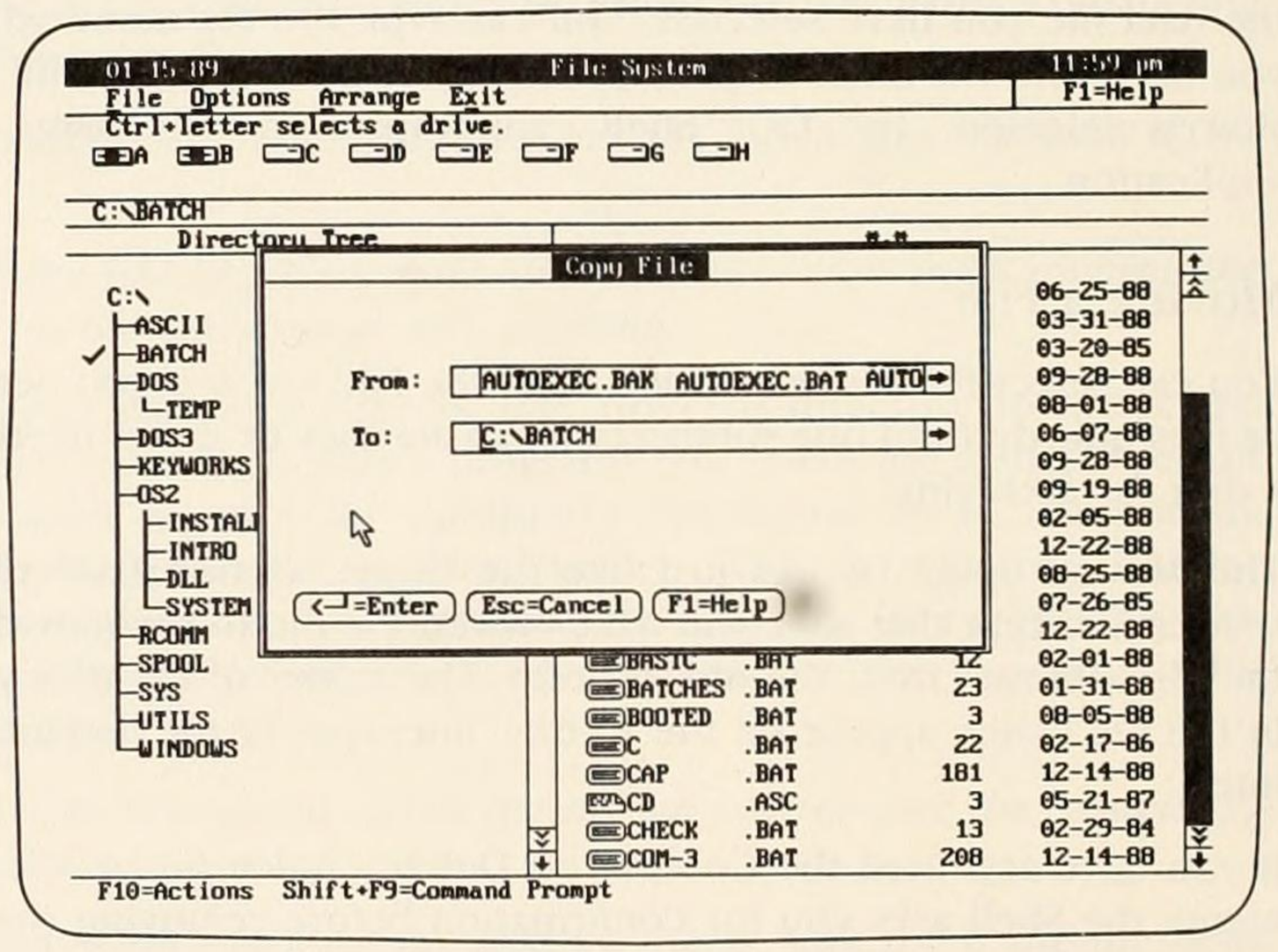


Fig. 16.17

The Copy File panel.

Cue:
Use the scroll techniques to scroll through the file names on the `From:` *line.*

The Copy File panel has two lines on which you can enter information. The `From:` line enables you to enter the name of the file or files to be copied. Because you have chosen the Select all option, each of the file names on the disk appear one after another on the line. Not all the names appear at once because the line has room for just a few. You can use the scroll bar to scroll through the listing, if you want.

Use the mouse or press Tab to move to the `To:` line and type the name of the destination drive or directory. If drive A: does not appear on the line, press Del to delete what appears there, type **A:**, and press Enter.

The light on drive B: goes on first, and then the light on drive A: goes on. The lights alternate for a minute or so as each file is copied from drive B: to drive A:. The name of the file being copied appears on the Copy File screen, along with the information that file 1 of x, 2 of x, or 3 of x, and so on, is being copied (where *x* is the number of files you selected). You always know how many files remain to be copied. Finally, a screen message tells you that DOS has successfully copied all the files.

Deleting a File

Before selecting the Delete... option, make sure that no files are selected that you don't want to delete. Highlight only the files you do want to remove, and then select the Delete... option from the File menu.

A panel pops up listing the selected files. Press Enter to start deleting the files. A confirmation panel appears for each file; you can choose to skip the current file and continue or to delete the file. You confirm the deletion of each file for all the files you selected. You can disable the confirmation step by using the File System Options menu described later in this chapter.

Renaming a File

To rename a file, choose Rename... from the File menu. A panel pops up with fields where you type the old file name and the new file name.

Changing a File's Attributes

File attributes are discussed in Chapter 7. You can change file attributes with the DOS Shell as well as with the DOS ATTRIB command. You can use the Shell Change attribute... option in batch mode to change several files one after another. You can also use the utility in global mode to change many files all at once in exactly the same way. You also can use the utility to change a single file. To change a file's attributes, follow these steps:

Step 1. *From the file listing, select the file or files you want, marking them with the space bar or mouse.*

Step 2. *Pull down the File menu and choose the Change attribute... selection.*

The Change Attribute menu appears. You can choose to change all selected files one at a time or simultaneously.

Step 3. *Choose either (1) One at a time or (2) All selected files at once. Press Enter.*

If you choose to change all selected files one at a time, each of the file names appears in turn on the `File:` line of the next panel. You select the desired attributes for each file. If you elect to change all selected files at one time, no file names appear, and you select attributes to be applied to all the files. Either way, the hidden, read-only, and archive attribute selections appear.

Step 4. *Choose the attributes you want to change.*

You can move the highlight by using the mouse or cursor keys. Press the space bar to toggle the attribute on or off. (If the attribute is on, pressing the space bar turns it off, and vice versa.)

Step 5. *When you have set the attributes as you want them for a given file, press Enter.*

The attributes are changed on the disk. If you have selected more than one file, the next file name appears in the panel, and you can repeat steps 4 and 5 until the last file is changed.

Viewing a File

Cue:
Use the View option to look at the contents of an ASCII file.

The View option enables you to look at the contents of ASCII files. You can page through the files by pressing the PgUp and PgDn keys.

To view a file, select it from the file list (either by using the mouse or by pressing the space bar when the file name is highlighted). Then pull down the File menu from the action bar and select View. The File View panel appears, as shown in figure 16.18. When you have finished viewing the file, press Esc to return to the File System.

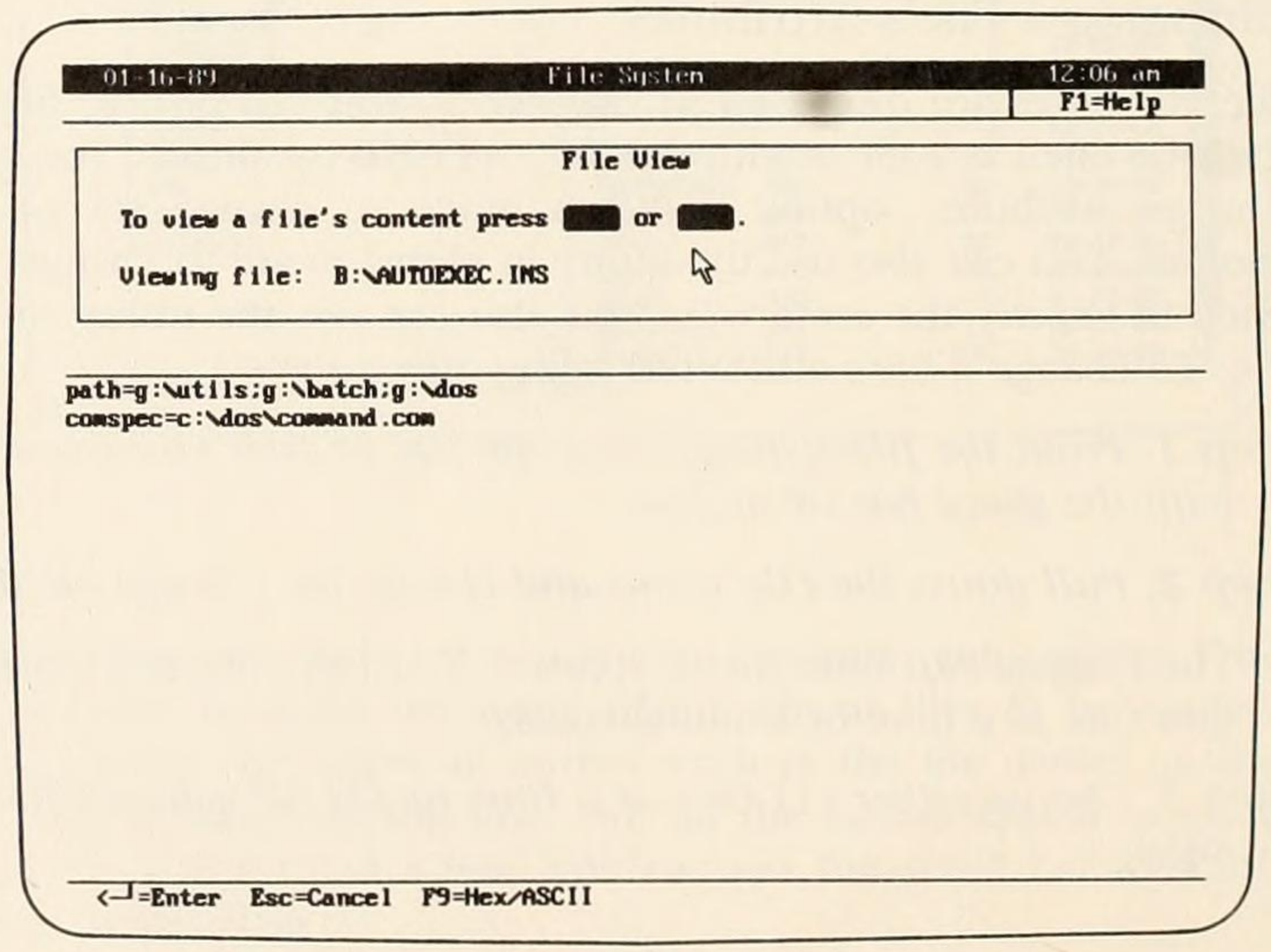


Fig. 16.18

The File View panel.

Creating a Directory

To make a new directory from the Shell File System, select Create directory... from the File menu. A panel appears, and you can type the name of the directory to be created.

Selecting All Files

To highlight all the files in the current directory (you may want to do this to copy or delete them all at one time), choose the Select all option from the File menu. All the files in the directory are highlighted.

Deselecting All Files

To reverse the preceding procedure and deselect all the files in the directory, use the Deselect all option from the File menu. All the highlighted files in the directory are deselected.

Using the File System Options Option

The File System's Options menu has three entries: Display options..., File options..., and Show information.... Each of these entries is described in the following sections.

Selecting Display Options

The Display options selection in the Options menu enables you to specify the order in which files are shown in the File System. The panel that appears when you select Display options lists the following items:

```
Name  Extension  Date  Size  Disk order
```

Select one of these items to sort your directory files by file root, file extension, creation or modification date, file size, or order in which the files appear on the disk. Use the cursor keys or mouse to select the option you want.

Selecting File Options

The File options selection enables you to specify how the Shell is to perform certain functions. You can select any or all of these options:

Option	*Description*
Confirm on delete	Requires the Shell to ask you before erasing a file. If you do not select this option, the Shell erases files without presenting a confirmation panel.

Option	*Description*
Confirm on replace	Causes the Shell to prompt you for confirmation before copying a file over another file with the same name.
Select across directories	Enables the Shell to choose files even if the files are in different directories.

Showing File Information

Figure 16.19 shows a typical Show Information panel, which displays four categories of information under the headings File, Selected, Directory, and Disk. The panel shows information about the file currently highlighted in the file listing.

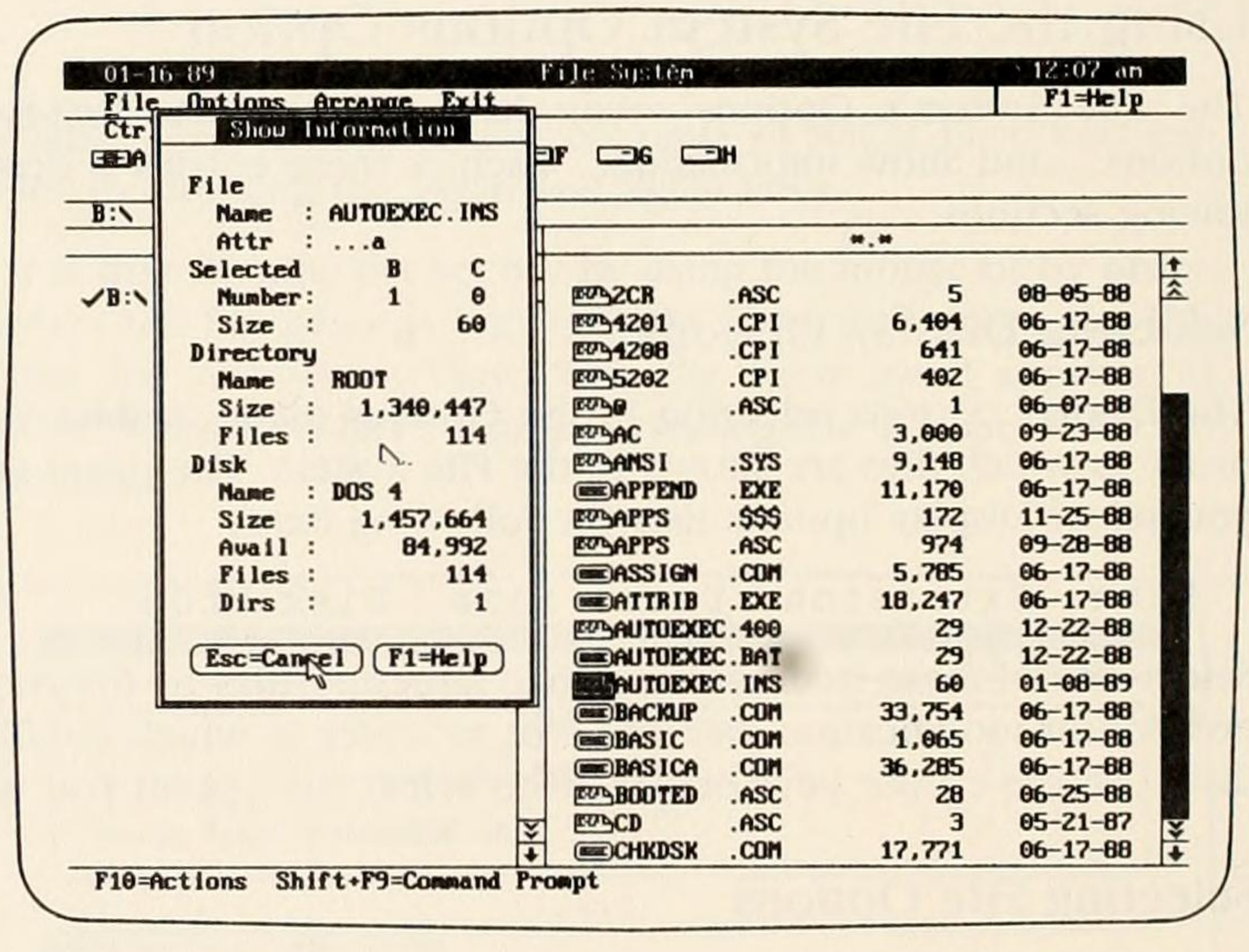


Fig. 16.19

The Show Information panel.

Reminder: *The Show Information panel displays data about a file, such as its directory and disk location.*

The File section of the Show Information panel displays the file's name (including the extension) and attributes. Each attribute set is indicated on the `Attr` line by a character: *r* for read-only, *h* for hidden, *s* for system, and *a* for archive bit. Any of these characters, none of them, or all four may appear on the line, depending on the file.

To the right of the Selected heading in the panel appear the names of the drives currently stored in the Shell's buffer. If more than one drive is "memorized,"

this entry and the Number and Size entries have two columns, one for each of the drives. The Number entry indicates how many files have been marked in the directory listing for that drive. The Size entry shows the total number of bytes taken up by the selected files in that drive.

The Directory section of the panel displays the name of the currently selected directory, the total size of all the files in that directory, and the total number of files in that directory.

The Disk section covers information about the disk drive itself. The Name entry is the label applied during formatting or with the LABEL command. The Size entry represents the total disk space, and the Avail entry signifies the free space available for new files. The Files entry shows the number of files stored on the disk. The last entry, Dirs, shows the number of subdirectories on the disk.

Although certain utility programs provide the same information as the Show Information panel, this panel gives DOS users easy access to all this data, in one place.

Using the File System Arrange Option

The File System Arrange menu is a powerful tool that enables you to choose files from a single file listing (the default option), from multiple windows that show file listings for two disks or directories, or from the current directory.

The Multiple File List Option

You use the Multiple file list option from the Arrange menu to show two file lists on-screen. To learn to use the multiple windows, follow these steps:

Step 1. *Make sure that the File System menu is on-screen, pull down the Arrange menu, and choose Multiple file list.*

The screen shows two panels, one above the other (see fig. 16.20). Each panel consists of a directory-tree section and a file-listing section. The current drive and directory listing appears in the top panel; the other buffered drive or directory appears in the bottom panel. Each panel displays fewer lines than its corresponding single-file listing would display.

Step 2. *Use the mouse or Tab key to move from one panel to the other.*

You can scroll through each of the listings and manipulate the directory tree of each individually.

Step 3. *If you want to select another drive for an active panel, move the mouse pointer to the drive symbol you want and click. Alternatively, hold the Ctrl key and press the letter associated with the desired drive.*

Each panel has its own drive-selector line, so you can change the drive for each panel.

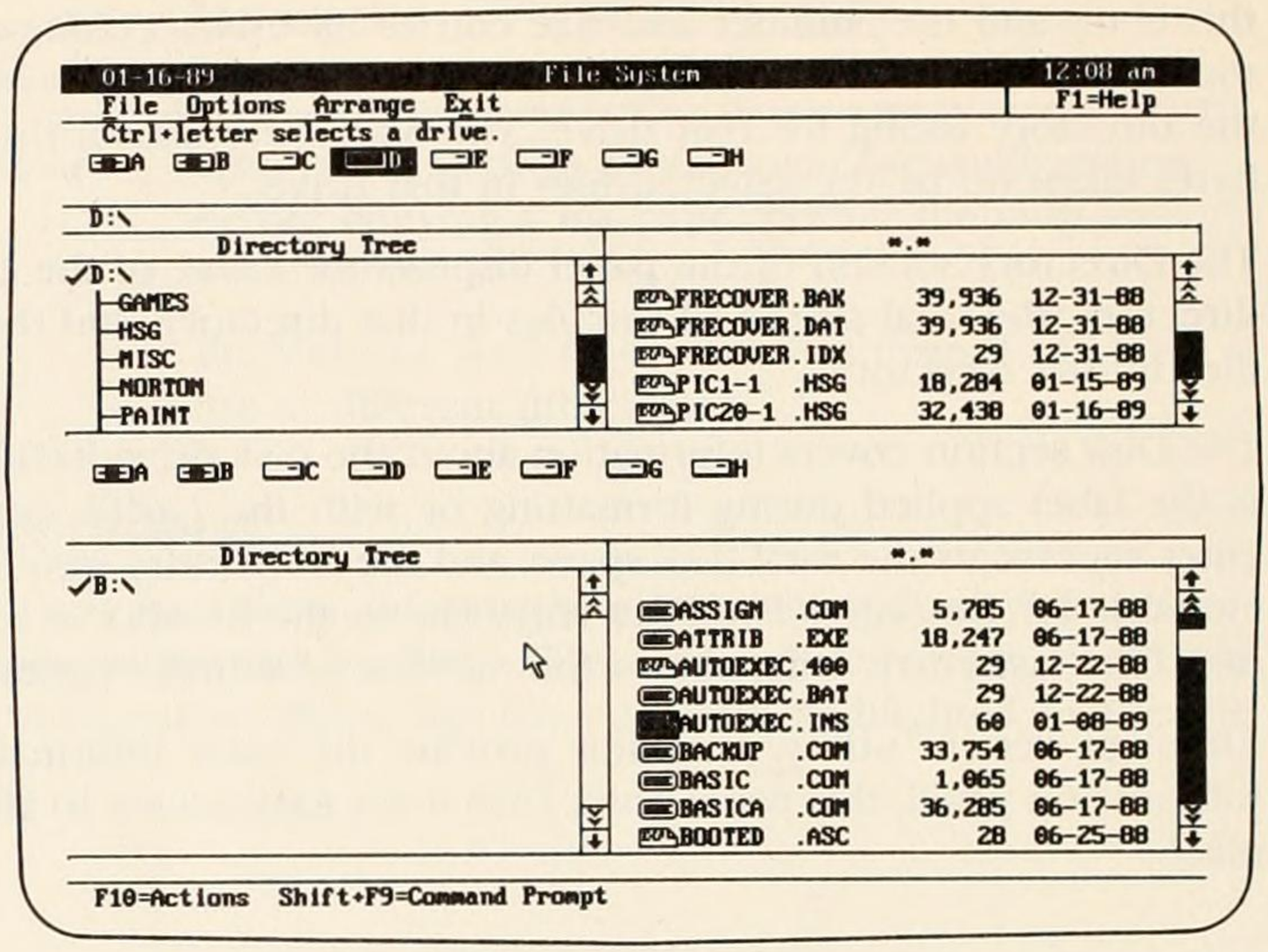


Fig. 16.20

The Multiple file list.

Step 4. *To return to the single-file list, pull down the Arrange menu and choose Single file list.*

The System File List Option

You can select the System file list option from the Arrange menu to show only the DOS system files for the currently selected disk.

Using the File System Exit Option

To exit the File System, access the File System Exit menu and select Exit File System. The Main Group/Start Programs menu appears. If you select Resume File System, the File System menu reappears.

Using Change Colors

Cue: *Experiment with colors until you find the one you like best.*

If you have a color monitor, you can select one of four combinations of colors in which to display the Shell. To choose one of these combinations, access the Change Colors utility from the Main Group. A screen showing the color combinations appears.

Use the right and left arrows to move through the four choices. When you see a combination you like, press Enter. The Shell saves that choice and uses it for the rest of the current session as well as all subsequent sessions.

Using the DOS Utilities Menu

The DOS Utilities menu lets you access several standard DOS commands. When each is selected, a panel appears in which you can type the standard parameters DOS requires for each of the commands. In all respects, these commands behave exactly the same as if you had typed them at the DOS prompt. These commands are available on the DOS Utilities menu for your convenience. Refer to the following chapters for descriptions of these commands:

Reminder:
The options on the DOS Utilities menu are the same as standard DOS commands.

DOS Utilities Option	*DOS Command*	*Chapter Reference*
Set Date/Time	TIME and DATE	8
Disk Copy	DISKCOPY	5
Disk Compare	DISKCOMP	5
Backup Fixed Disk	BACKUP	7
Restore Fixed Disk	RESTORE	7
Format	FORMAT	5

Installing DOS V4 and the Shell with SELECT

Cue:
Use the SELECT utility to install DOS and the Shell.

You can use the SELECT utility that comes with DOS to do more than install the Shell; you can use it to install DOS also. SELECT can create diskettes you can use to boot from your floppy disk drive (much like the DOS V3 SELECT utility); SELECT also can copy DOS to your hard disk. The SELECT utility is not completely new; the utility in DOS V3 was used to prepare a diskette for use with various country codes and keyboard combinations. If you had no need to change country codes, however, you probably never had occasion to use SELECT. In versions of DOS before V4, SELECT worked only with diskettes.

When you create a bootable disk with SELECT, the program asks you to make some decisions. These choices affect how efficiently your computer uses DOS as well as which DOS V4 features are available to you. The decisions you make are not permanent; you can change them later.

You cannot use the SELECT utility just to install the Shell; you have to install DOS and the Shell. The following instructions explain how to install DOS and the Shell on your hard disk or to create a bootable floppy disk.

To start SELECT, insert the DOS V4 disk labeled "Install" in drive A: and boot your computer. If the computer is already turned on, press Ctrl-Alt-Del. If you use 5 1/4-inch diskettes, remove the Install disk when prompted, insert the

SELECT disk, and press Enter. If you use 5 1/4-inch diskettes, you will have to reinsert the Install disk when prompted.

Cue:
Know the special key assignments used by the SELECT utility.

The opening screen appears, which displays the copyright notice and asks you to press Enter to continue or Esc to cancel. If you press Enter, the Welcome screen appears, which lets you know how many diskettes you need to complete the installation. Press Enter to continue with the installation. The Introduction screen is next, which reports the special key assignments used by the program (see table 16.1).

Table 16.1
SELECT Key Assignments

Key	*Use*
Enter	Proceed to the next step
Esc	Cancel the current display
Tab	Move to the next entry field
PgUp or PgDn	Scroll information one page at a time
Up or Down Arrow	Move the highlight bar to the preceding or next item
F1	View help information
F3	Exit SELECT
F9	View key assignments while viewing help
Left or Right Arrow	Scroll data fields horizontally to the left or right

Because SELECT is menu-oriented and contains complete help instructions, the SELECT menus are not explained here. Instead, a summary of the steps you follow to install DOS and the DOS Shell are provided.

Choosing a Configuration

Reminder:
Select the configuration that gives you the most features without limiting system memory.

When you choose a configuration, you provide DOS with information it uses to set up your system. You learn to use the functions and parameters to specify a system configuration later in this chapter. In the Specify Function and Workspace screen, you are asked to choose from three options:

1. Minimum DOS function; maximum program workspace
2. Balance DOS function with program workspace
3. Maximum DOS function; minimum program workspace

These options let you specify whether you want to set up your system so that some of DOS's specialized functions are enabled, or whether you want those functions disabled.

The specific functions you can enable include utility programs that make your system run a tiny bit faster. If you do not enable these DOS functions, more of your computer's memory is available to run larger programs. Additional memory is also spared for programs like 1-2-3, which functions better with more memory.

Choosing option 1, Minimum DOS function; maximum program workspace, provides you with the maximum amount of free memory. Because DOS V4 uses about twice as much memory as DOS V3.3 (128K), you may need to select option 1 if you want to run some of your memory-hungry programs. Selecting option 1 leaves about 564K of memory free for your applications.

Option 2, Balance DOS function with program workspace, enables some of the special DOS functions and leaves a little less memory free—approximately 548K. Select this option if you don't plan to load other programs that permanently grab their own share of memory.

Option 3, Maximum DOS function; minimum program workspace, enables all the special DOS features, leaving about 534K of free memory. You may have enough memory to run your applications, but you should run each to make sure that the fully featured DOS does not take too much memory. For example, load 1-2-3 and your largest worksheet and see whether you have enough memory.

Remember, you can add DOS functions at any time by running SELECT again or by adding statements to the special CONFIG.SYS file, described in Chapter 14.

Reminder: *You can rerun SELECT to add DOS functions or modify the CONFIG.SYS file.*

After deciding on the special functions to be installed, you choose a predefined country (the default is `United States`) and a predefined keyboard (the default is `None`, or no special keyboard). You can accept the predefined country and keyboard or specify a different country and keyboard.

Choosing Drive and Printer Information

The next two SELECT screens enable you to specify where you want to install DOS V4. You can choose between drive C (usually the hard disk) and drive A (the first floppy disk). If you select drive A, you create a bootable floppy disk; if you select drive C, you install DOS on your hard disk.

Reminder: *Install DOS V4 on either drive A or C, and select up to seven printers.*

SELECT then asks how many printers you have. You can specify from none up to seven. You can specify only as many printers as you have ports. The ports are LPT1, LPT2, LPT3, COM1, COM2, COM3, and COM4. Most users specify one or two printers: a letter-quality and a draft-quality printer. You choose each printer from the list SELECT displays.

Once you have specified your printers, SELECT gives you the chance to accept or change the current configuration and continue with the installation of DOS V4. The items you can change include code-page switching, extended display support, GRAFTABL support, GRAPHICS print screen support, DOS Shell, and RAM disk (these items are described in Chapters 14 and 15).

SELECT then prompts you for the time and date and copies the operating system files to the selected disk. SELECT also creates the batch file DOSSHELL.BAT, which contains the commands needed to load the Shell properly. You load the Shell after the installation is complete by typing **DOSSHELL** at the A> or C> prompt.

Understanding a Sample DOSSHELL.BAT File

Here is a typical DOSSHELL.BAT file. The SELECT utility creates one similar to this for you. Your file may differ slightly depending on the options you selected and the changes you made to this file.

```
(1) @ECHO OFF
(2) C:
(3) CD \DOS
(4) SHELLB DOSSHELL
(5) IF ERRORLEVEL 255 GOTO END
(6) :COMMON
(7) @SHELLC/MOS:PCIBMDRV.MOS/TRAN/COLOR/DOS/MENU/MUL
    /SND/MEU:SHELL.MEU/CLR:SHELL.CLR/PROMPT/MAINT/EXIT/SWAP/DATE
(8) :END
(9) CD \
```

Each of the lines in this batch file is numbered for easy reference. The line numbers are not part of the batch file, however, and should not be included if you copy this file or create one of your own from scratch.

Following are explanations of each line of the sample DOSSHELL.BAT file.

Line 1: Turns off the echoing of commands so that they are not displayed on-screen as DOS carries them out.

Line 2: Changes the current drive to drive C in case you happen to be using some other drive at the time you call DOSSHELL.BAT.

Line 3: Changes the current directory to \DOS, where the programs you need to run the Shell are stored.

Line 4: Summons a program called SHELLB.COM, which loads some key functions into memory. These functions take up less than 4K of DOS memory. SHELLB.COM also performs two other tasks that enable the remaining Shell programs to load. First, SHELLB.COM checks for any errors that might prevent the Shell from running properly. If any errors are found, a value of *255* is loaded into the special DOS memory register that stores the current ERRORLEVEL. Second, SHELLB.COM notes the name of the batch file you want to use to call up the DOS Shell. The default name is DOSSHELL. In network environments, customized batch files can be used for each note using the Shell on the same server.

Line 5: Checks to see whether SHELLB.COM found any errors by checking whether ERRORLEVEL equals 255. If an error was found, control passes to line 8 (:END), and the batch file terminates. If this happens, you may have to run the SELECT program again, making sure you correctly define your system.

Line 6: Marks the beginning of the portion of the batch file that calls the SHELLC.COM program. When you run a program from the DOS Shell or leave the Shell to access the DOS prompt, DOS returns to this line in the batch file to reload the Shell. You must be careful to specify to SHELLB.COM the name of the file you want to use; you also must use only the :COMMON label to mark this point in the batch file. If you don't specify the name of the file, DOS does not know where to look. If you use some label other than :COMMON, DOS cannot find the correct starting place.

Caution: *If you do not provide SHELLB with a batch-file name or use a label other than :COMMON, the Shell does not work.*

Line 7: Does most of the work in setting up the DOS Shell. A detailed examination of this line appears in the following section.

Line 8: Marks the end of the section of the batch file that calls SHELLC.COM.

Line 9: Changes the current subdirectory to the root directory.

Customizing the Shell

This section first examines the options listed in line 7 of the sample DOSSHELL.BAT file and then discusses those options not included in the sample batch file. Line 7 of the sample file calls the SHELLC.COM program and specifies a list of options that customize the Shell. When including options in your DOSSHELL.BAT file, remember that no spaces can precede the slashes for the options in the SHELLC line.

Cue: *Do not include spaces in front of the slashes when specifying Shell options.*

The /MOS: Option

The /MOS: option enables you to specify the mouse driver to be used with the DOS Shell. Include this option only if you use a mouse with the Shell. The IBM mouse, for example, uses the PCIBMDRV.MOS driver. The Shell also provides the PCMSDRV.MOS driver for the Microsoft serial mouse and the PCMSPDRV.MOS driver for the Microsoft parallel mouse. One of these three drivers should work with your mouse; each requires only 1K of memory. You must specify one of these three drivers if you have a mouse; other drivers do not work with the Shell. You should find these drivers in the same directory as the Shell programs. DOS deactivates these drivers when the Shell is not active so that these drivers do not conflict with any other mouse drivers you may use.

The /TRAN Option

Cue:
Use the /TRAN option if you want the Shell to use as little RAM as possible.

The /TRAN option activates the transient mode of the Shell. In *resident* mode, the DOS Shell and the buffers it uses are loaded into memory and stay there. The buffers use all available memory (unless you limit the buffer size with the /B: option, described later in this chapter). If the Shell operates in resident mode, the Shell works faster because DOS doesn't have to access the hard disk. Because the entire Shell is available in memory for instant access in resident mode, however, you have less free memory for other uses. For more information on buffers, refer to Chapter 14.

In *transient* mode, the Shell uses only 3.5K of memory and stores the other information it needs on the hard disk. Because you run other programs from the Shell (you do not exit the Shell before starting another program), limiting the amount of RAM the Shell uses is a good idea if you want memory available for such RAM-hungry programs as Ventura Publisher.

The /COLOR Option

If you do use a color monitor, the /COLOR option activates the Change Colors utility in the Main Group menu. If you omit this option from DOSSHELL.BAT, you cannot change the colors of the Shell. The Change Colors selection still appears in the Main Group menu.

The /DOS Option

Reminder:
You can add the /DOS option (or any other option) to DOSSHELL.BAT at any time.

The /DOS option makes the File System selection available on the Main Group menu. If you omit the /DOS option from DOSSHELL.BAT, you cannot perform any File System functions such as copying files and viewing directories. You can always modify DOSSHELL.BAT to include the /DOS option later.

The /MENU Option

The /MENU option activates the Start Programs main menu. If you don't specify this option, you can access only the File System menu. (If you don't specify the /DOS option or the /MENU option, you cannot do anything.)

Reminder:
Use the /MENU or /DOS option, or both, or you cannot do anything with the Shell.

The /MUL Option

When the /MUL option is specified, the Shell keeps track of the information obtained from the last two drives accessed. Having the Shell keep track of this information can speed up File System operations such as displaying the directory tree and listing files, because the Shell doesn't have to access the disk to recall the information.

The /SND Option

The /SND option activates the sounds you hear when working with the Shell (it does not control the sounds made by programs you start from the Shell). Unless you work in a "quiet zone," you may want the added feedback that your computer's various beeps can provide.

The /MEU: Option

The /MEU: option enables you to define the name of the menu file used for the Main Group menu. The default file is SHELL.MEU. If you want to use the SHELL.MEU file, you don't have to include the /MEU: option at all.

Cue:
If you want to use the default Shell menu, you don't need the /MEU: option.

If you want to create a new Main Group menu, specify a file other than SHELL.MEU. Suppose that you deleted the Change Colors utility and also want to remove it from the Main Group menu. To do this, create a new subgroup menu that has the headings you want and provide the menu with a file name (refer to "Adding a Group" earlier in this chapter). Edit the DOSSHELL.BAT file so that the /MEU: option includes the new menu file name.

The /CLR: Option

The /CLR: option enables you to specify the name of the file containing the DOS Shell color scheme you have selected. The default file is SHELL.CLR. If you want to use the default file, you don't have to include the /CLR: option at all. If you want, you can write different versions of DOSSHELL.BAT, each specifying a

different color-scheme file. You create color schemes using the Change Colors option on the Main Group menu (refer to "Using Change Colors" earlier in this chapter).

The /PROMPT Option

Cue: *Include the /PROMPT option for quick access to DOS.*

The /PROMPT option enables you to switch from the Shell to the DOS command prompt so that you can carry out functions or run other programs. The /PROMPT option enables the Shift-F9 key and the Command Prompt option on the Main Group menu. To return to the Shell after you have accessed the DOS prompt, type **EXIT** at the DOS prompt.

If you omit the /PROMPT option, you are "stuck" inside the Shell with no means of quick access to DOS (other than exiting from the Shell entirely.)

The /MAINT Option

The /MAINT option provides access to the various action-bar commands that appear in the submenus (such as Add... and Change...). If you omit /MAINT, you can use only the commands found in the group listings themselves. Each of the Main Group programs and utilities are accessible if you omit /MAINT, but you cannot add or change programs and groups. In the File System, you can view directories and the directory tree if you omit /MAINT, but you cannot copy or delete files, create directories, and so on. In short, including /MAINT enables you to change the Shell; omitting /MAINT prevents you from accessing commands that change the Shell.

The /EXIT Option

Reminder: *Include either the /EXIT or /PROMPT option, or you can return to DOS only by rebooting the computer.*

The /EXIT option activates the action-bar selection that enables you to unload the Shell and return complete control to DOS. If you do not specify /EXIT, you are "trapped" in the Shell. If you also do not specify the /PROMPT option, you cannot access the DOS prompt unless you reboot the system.

The /SWAP Option

The Shell can maintain a file that stores the buffered information about files and directories used by the File System. If you specify the /SWAP option, DOS stores this information on disk in a temporary file. This file helps speed up the reloading of the Shell when you exit from a program. Like the /MUL option, /SWAP can speed up the operation of the Shell; you can specify both of these options if you want.

The /DATE Option

When you specify the /DATE option, the date and time are displayed in the DOS Shell title bar and are updated every minute. You may not want to use the /DATE option if you have installed the Shell on a computer with a nonfunctional clock.

Reminder: *Use the /DATE option to display the time and date in the Shell title bar.*

The Shell options described in the following sections do not appear in the sample batch file, but you may want to include them in your own DOSSHELL.BAT file.

The /ASC: Option

The DOS Shell Associate... function enables you to correlate various file extensions with particular applications programs. You can use Associate... to pair the .DOC extension with your word processing program, for example, or the .WKQ extension with your Quattro spreadsheet program. When you select a file with an "associated" extension, the Shell automatically loads the correlated application for you.

DOS stores the links between file extensions and application programs in a file called SHELL.ASC. If your computer has several users, you may want to store each user's associations in a different file. (You create the association file by selecting File System from the Main Group menu, selecting the File option, choosing Associate..., and filling out the displayed screen.) Use the /ASC: option to specify the name of another association file.

The /ASC: option allows more flexibility in loading programs. You don't have to know which file contains the program you want to start; you can just select the file you want to access and let DOS find and start the associated program file.

The /B: Option

If you use the /B: option, you can limit the amount of memory that the Shell uses in resident mode for its buffers. Specify the size in whole kilobytes. If you want to limit to 8K the amount of RAM the Shell uses for buffers, for example, specify /B:8. If you don't use the /B: option, DOS uses all available memory for buffers. Omitting the /B: option usually does not cause problems.

The /CO1, /CO2, and /CO3 Options

Cue: *Experiment with the /CO1, /CO2, and /CO3 options to see which works best with your system.*

Use the /CO1, CO2, or /CO3 option to specify the video-graphics mode the Shell is to use. The mode you choose must be compatible with your display

adapter and monitor. You can use some of these options with more than one type of monitor. Try several to see which are compatible with your system and use the one you prefer. The options and their associated modes are listed in the following chart:

Option	*Mode*
/CO1	EGA 640×350 16-color mode; produces an 80-column, 25-line display (technical people know this as video mode 10H)
/CO2	VGA 640×480 black-and-white mode; produces an 80-column, 30-row display (in technical terms, this is video mode 11H)
/CO3	VGA 640×480 16-color mode; produces an 80-column, 30-row display (also known as video mode 12H)

The /TEXT Option

The /TEXT option instructs the Shell to use text mode for CGA monitors or Monochrome Display Adapter (MDA) monitors. These video modes are 3 and 7, respectively.

The /COM2 Option

Ordinarily, the Shell uses the COM1 serial port for the mouse. If you include the /COM2 option, the Shell uses COM2 for the mouse.

The /LF Option

Caution: *Using the /LF option to switch mouse buttons in the Shell may lead to confusion in other programs.*

The /LF option changes the mouse driver's configuration so that you use the right button instead of the left button with the Shell. This option is useful for left-handed people. Because this option controls the action of the mouse within the Shell, and not in programs started from the Shell, switching mouse buttons may be confusing.

Summary

Important points covered in this chapter include the following:

- ❏ You can choose to "balance" your system between maximum DOS function and maximum memory space.
- ❏ You can change the DOS Shell by altering the DOSSHELL.BAT file.

- ❑ Many of the features of the Shell can be disabled or activated, as you prefer.
- ❑ Customized shells can be prepared for different users.
- ❑ The Shell's menus are arranged in tree fashion.
- ❑ You can add or change the program names that appear on the Shell menu.
- ❑ You can help users by adding customized prompt and help panels to the menu choices.
- ❑ The DOS Shell File System enables you to copy or delete multiple files using the menus.
- ❑ The File System enables you to change the attributes of files.
- ❑ You can choose the order in which the File System displays files.
- ❑ Files can be erased automatically after they are copied to another disk or directory.
- ❑ You can associate an application with the file extensions its data files use. As a result, the application is loaded automatically when you attempt to open the data file.
- ❑ You can specify whether the File System asks for confirmation before deleting or overwriting files.
- ❑ The Shell provides an extensive Show Information listing.
- ❑ You can view the directories of more than one disk or subdirectory at one time by using the Multiple file list option.
- ❑ You can change the screen colors in which the Shell appears.
- ❑ Several DOS commands, such as DISKCOPY, TIME, and Date, are available on the DOS Utilities menu.
- ❑ The DOS Shell is installed with the SELECT program.

Part V

Command Reference

Introduction to the Command Reference

This Command Reference includes all DOS commands. DOS versions with which the command can be used follow the command name. Then, you will see either the term *internal*, indicating a command built into DOS, or the term *external*, indicating a disk-resident command.

The purpose of the command is explained, followed by the syntax required to invoke the command. Any switches used with the command are given with information about their correct use in the command line. Then any applicable exit codes are listed.

Next is a reference to the chapter in the main section of the book where the command is discussed more fully. A set of rules governing the use of the command is given. An example is shown for some commands, where it may be helpful.

The Notes contain additional comments, background information, or suggestions for using the command effectively. This section also lists differences in use of the command for specific machines and indicates computers for which the command may not be available.

The Messages section is an alphabetical listing of the messages produced by the command. There are three types of messages:

- INFORMATION — This message simply informs the operator that an activity is taking place—or the message may prompt the operator for a response.
- WARNING — This message warns the operator of a possible problem. The command continues to function.
- ERROR — This message indicates that an error has occurred. The error may be a command error, caused by the way the information on the command line was entered. Or the error message may occur during an operation, indicating a disk or other hardware error. The command terminates after an error message.

Notation

The following notation represents a file specification as it occurs in a command line:

d:path\filename.ext

d: names the disk drive that holds the file, and *path* is the directory path to that file. *filename* is the root name of the file, and *.ext* is the file name extension.

If any part of this notation does not appear in the file specification under Syntax, the omitted item is not allowable with the cited command. For example, the notation *d:filename.ext* indicates that path names are not allowed in the command. In most cases, a device name may be substituted for a full file specification.

Any part of a notation under Syntax appearing **in bold type face**, is mandatory and must be entered to invoke the command. Any portion of the notation appearing *in italic type face*, is optional and is entered only when appropriate.

Notation for External Commands and Batch Files

Starting with DOS V3, external commands, which are simply program files, and batch files residing in subdirectories can be executed just as program and batch files can be executed from different disk drives. The syntax line shows a *c* added to the disk drive name and path, where this type of command is used. A typical notation is as follows:

*dc:pathc***command**

dc: is the name of the disk drive holding the command; *pathc*\ is the directory path to the command, and **command** is the name of the program or batch file to execute.

Use this notation as a syntactical model either for an external, disk-resident command, not part of COMMAND.COM, or to invoke a batch file command.

For this command notation, the following rules apply:

1. If you fail to name a disk drive *dc:* for the command, DOS will search for the command on the current disk drive.
2. If you do not designate a path *pathc*\, DOS will search for the command on the current directory of the current disk—or the current directory of the disk drive you specified.
3. If you do not give a drive and a path name, DOS searches the current directory of the current disk for the command. Not finding it there, DOS next searches the list of paths specified by the PATH command. If DOS does not find the command after searching the path, DOS displays the error message `Bad command or file name` and gives the DOS system prompt, which is usually `A>` or `C>`.

Using Upper- and Lowercase Letters

Commands, as well as all parameters and switches typed with commands, may be entered in either upper- or lowercase. The exceptions are with FIND and the batch subcommands; for these commands, the use of upper- or lowercase for certain parameters may be important. You will need to observe the syntax lines provided with the commands for such instances.

DOS Messages

DOS messages fall into two groups: *general* and *device error* messages. The larger group, general DOS messages, will be listed first, followed by the device error messages.

The wording of error messages for your implementation and version of DOS may differ from those shown here. Sometimes the variations may be as slight as punctuation and capitalization. At other times, the entire content of the message may differ. If you see a message that you cannot locate in this guide, refer to the DOS manual for your computer.

General DOS Messages

The following messages may appear when you are starting DOS or using your computer. Messages associated with starting DOS are marked as start-up. Most start-up errors mean that DOS did not start and that you must reboot the system. Most of the other error messages mean that DOS terminated, or aborted, the program and returned to a system prompt, such as `A>`, or `C>`.

`Bad command or filename`

ERROR: The name you entered is not valid for invoking a command, program, or batch file. The most frequent causes are the following: (1) you misspelled a name; (2) you omitted a needed disk drive or path name; or (3) you gave the parameters without the command name, such as typing **myfile** instead of **ws myfile**, omitting the **ws** specifying WordStar.

Check the spelling on the command line. Make sure that the command, program, or batch file is in the location specified by disk drive and directory path. Then try the command again.

`Bad or missing Command Interpreter`

ERROR (start-up): DOS cannot find the command interpreter, COMMAND.COM. DOS does not start.

If you are starting DOS, this message means that COMMAND.COM is not on the DOS start-up disk or that a version of COMMAND.COM from a previous DOS is on the disk. If you used the SHELL directive of CONFIG.SYS, the message means that the directive is improperly phrased or that COMMAND.COM is not where you specified. Place another diskette containing the operating system in the floppy disk drive, then reset the system. After DOS has started, copy COMMAND.COM to the original start-up disk so that you can boot from that disk.

If this message appears while you are running DOS, several possible causes exist: (1) COMMAND.COM has been erased from the disk and directory you used to start DOS, (2) a version of COMMAND.COM from a previous DOS has overwritten the desired version, or, (3) the `COMSPEC=` entry in the environment has been changed. You must restart DOS by resetting the system.

If resetting the system does not solve your problem, use a copy of your DOS master diskette to restart the computer. Copy COMMAND.COM from this diskette to the offending disk.

`Bad or missing filename`

WARNING (start-up): You requested DOS to load a device driver that could not be located, an error occurred when the device driver was loaded, or a break address for the device driver was beyond the bounds of the computer's RAM. DOS will continue its boot, but will not use the device driver `filename`.

If DOS loads, check your CONFIG.SYS file for the line `DEVICE=filename`. Make sure that the line is spelled correctly and that the device driver is where you specified. If this line is correct, reboot the system. If the message appears again, copy the file from its original diskette to the boot diskette and try rebooting DOS. If the error persists, the device driver is defective and you should contact the dealer or publisher who sold you the driver.

`Batch file missing`

ERROR: DOS could not find the batch file it was processing. The batch file may have been erased or renamed. With DOS V3.0 only, the diskette containing the batch file may have been changed. DOS aborts the processing of the batch file.

If you are using DOS V3.0 and you changed the diskette containing the batch file, restart the batch file without changing the diskette. You may need to edit the batch file to avoid changing diskettes.

If the batch file includes a RENAME command, causing the originating batch file's name to be changed, edit the batch file in such a way that it will not rename itself when processed again.

If the file was erased, re-create the batch file from its backup file if possible. Edit the file to ensure that the batch file does not erase itself.

`Cannot load COMMAND, system halted`

ERROR: DOS attempted to reload COMMAND.COM, but the area where DOS keeps track of available and used memory was destroyed, or the command processor was not found in the directory specified by the `COMSPEC=` entry. The system halts.

This message indicates either that COMMAND.COM has been erased from the disk and directory you used when starting DOS, or that the `COMSPEC=` entry in the environment has been changed. Restart DOS. If it does not start, the copy of COMMAND.COM has been erased. Restart DOS from the original master diskettes and copy COMMAND.COM to your other disk.

Another possible cause for this message is that an errant program corrupted the memory allocation table, where DOS tracks memory. Try running the program that was in the computer when the system halted. If the problem recurs, the program is defective. Contact the dealer or publisher who sold you the program.

`Cannot start COMMAND, exiting`

ERROR: DOS was directed to load an additional copy of COMMAND.COM, but could not. Either the `FILES=` command in CONFIG.SYS is set too low, or you do not have enough free memory for another copy of COMMAND.COM.

If your system has 256K or more and `FILES` is less than 10, edit the CONFIG.SYS file on your start-up diskette and use **FILES=15** or **FILES=20**; then reboot the computer.

If the problem occurs again, either your computer has insufficient memory or too many resident or background programs are competing for memory space. Restart DOS without unneeded resident or background programs. You may need to eliminate unnecessary device drivers or RAM-disk software. The next alternative is to increase the amount of RAM in your system.

`Configuration too large`

ERROR (start-up): DOS could not load itself because you specified too many FILES or BUFFERS in your CONFIG.SYS file. This problem should occur only on systems with the minimum amount of memory required to operate DOS.

Restart DOS with a different diskette and edit the CONFIG.SYS file on your boot diskette, lowering the number of FILES and/or BUFFERS. Restart DOS with the edited diskette. Another alternative is to increase the RAM in your system.

`Current drive is no longer valid`

WARNING: You set the system prompt to `PROMPT %p`. At the system level, DOS attempted to read the current directory for the disk drive and found the drive no longer valid.

If the current disk drive is set for a floppy disk, this warning appears when no diskette is in the drive. DOS reports a `Drive not ready` error. Give the **F** command to fail (which is the same as **A** for abort) or the **I** command to ignore the error. Then insert a floppy diskette into the disk drive.

The invalid drive error can also occur with a currently networked or a SUBST disk drive that has been deleted or disconnected. Simply change the current disk to a valid disk drive.

`Disk boot failure`

ERROR (start-up): An error occurred when DOS tried to load itself into memory. The diskette contained IO.SYS and MSDOS.SYS, but one of the two files could not be loaded. DOS did not boot.

Try again to start with the DOS diskette. If the error recurs, try booting DOS from a diskette you know to be good, such as a copy of your DOS master diskette. If this action fails, you have a hardware disk drive problem and should contact your dealer.

If you have a hard disk, type **SYS C:** and copy COMMAND.COM to the root directory to put these files on the hard disk.

`Divide overflow`

ERROR: A program attempted to divide by zero. DOS aborts the program. Either the program was incorrectly entered, or it has a flaw in logic. With well-written programs, this error should never occur. If you wrote the program, correct the error and try again. If you purchased the program, report the problem to the dealer or publisher.

This message can also appear when you attempt to format a RAM disk. Make sure that you are formatting the correct disk and try again.

`Error in COUNTRY command`

WARNING (start-up): The COUNTRY directive in CONFIG.SYS either is improperly phrased or has an incorrect country code or code page number. DOS continues its start-up, but must use the default information for the COUNTRY directive.

After DOS has started, check the COUNTRY line in your CONFIG.SYS file. Refer to Chapter 15 to verify that the directive is correctly phrased (using commas between country code, code page, and the COUNTRY.SYS file) and that any given information is correct. If you detect an error in the line, edit the line, save the file, and restart DOS.

If you can find no error, restart DOS. If the message reappears, edit your CONFIG.SYS file. Reenter the COUNTRY directive and delete the former COUNTRY line. The line may contain nonsense characters that DOS sees but are not apparent to your text editing program.

`Error in EXE file`

ERROR: DOS detected an error while attempting to load a program stored in an .EXE file. The fault is in the relocation information DOS needs to load the program. This problem can occur if the .EXE file has been altered in any way.

Restart DOS and try the program again, this time using a backup copy of the program. If the message reappears, the program is flawed. If you are using a purchased program, contact the dealer or publisher. If you wrote the program, use LINK to produce another copy of the program.

`Error loading operating system`

ERROR (start-up): A disk error occurred while DOS was loading itself from the hard disk. DOS does not boot.

Restart the computer. If the error occurs after several tries, restart DOS from the floppy disk drive. If the hard disk does not respond—that is, you cannot run DIR or CHKDSK without getting an error—you have a problem with the hard disk. Contact your dealer. If the hard disk does respond, use the SYS command to put another copy of DOS onto your hard disk. You may need to copy COMMAND.COM to the hard disk also.

`EXEC failure`

ERROR: DOS encountered an error while reading a command or program from the disk, or the CONFIG.SYS `FILES=` directive has too low a value.

Increase the number of FILES in the CONFIG.SYS file of your start-up disk to 15 or 20; then restart DOS. If the error recurs, you may have a problem with the disk. Use a backup copy of the program and try again. If the backup copy works, copy it over the offending copy.

If an error occurs in the copying process, you have a flawed diskette or hard disk. If the problem is a diskette, copy the files to another diskette and reformat or retire the original. If the problem is the hard disk, immediately back up your entire hard disk, then run RECOVER on the offending file(s) only. RECOVER is a potentially dangerous command; (refer to this command in the Command Reference). If the problem persists, your hard disk may have a hardware failure.

```
File allocation table bad, drive d
Abort, Retry, Fail?
```

WARNING: DOS encountered a problem in the File Allocation Table (FAT) of the disk in drive `d`. Enter **R** for Retry several times. If this does not solve the problem, use **A** for Abort.

If you are using a diskette, attempt to copy all the files to another diskette and then reformat or retire the original. If you are using a hard disk, back up all files on the disk and reformat the hard disk. The disk is unusable until reformatted.

```
File creation error
```

ERROR: A program or DOS attempted to add a new file to the directory or replace an existing file, but failed.

If the file already exists, use the ATTRIB command to check whether the file is marked as read-only. If the read-only flag is set and you want to change or erase the file, use ATTRIB to remove the read-only flag, then try again.

If the problem is not a read-only flag, run CHKDSK without the /F switch to determine whether the directory is full, the disk is full, or some other problem exists with the disk. This error message can result when a wild card is substituted for a file name.

```
File not found
```

ERROR: DOS could not find the file you specified. The file is not on the correct diskette or in the correct directory—or you misspelled the disk drive name, path name, or file name. Check these possibilities and try the command again.

```
filename device driver cannot be initialized
```

WARNING (start-up): In CONFIG.SYS, either the parameters in the device driver file name are incorrect, or the DEVICE line is in error. Check for incorrect parameters or for phrasing errors in the DEVICE line. Edit the DEVICE line in the CONFIG.SYS file, save the file, and restart DOS.

```
General failure
```

This is a catchall error message not covered elsewhere. The error usually occurs when you use an unformatted diskette or hard disk, or when you leave the disk drive door open.

```
Incorrect DOS version
```

ERROR: The file that contains the command you just entered is from a different version of DOS.

Obtain a copy of the command from the correct DOS version (usually from your copy of the DOS master diskette) and try the command again. If the disk you are using has been updated to hold new versions of the DOS programs, copy those versions over the old ones.

```
Insert disk with \COMMAND.COM in drive d
and strike any key when ready
```

INFORMATION and WARNING: DOS needed to reload COMMAND.COM but could not find it on the start-up disk.

If you are using diskettes, probably the diskette in drive A: has been changed. Place in drive A: a diskette holding a good copy of COMMAND.COM and press a key.

If you have started DOS from a hard disk, either COMMAND.COM has been erased or your COMSPEC has been set incorrectly. Type SET and press Enter. Look for the COMSPEC environment variable. You may see:

```
COMSPEC=C:\DOS\COMMAND.COM
```

Check to see if COMMAND.COM is in the directory specified by COMSPEC. If it is not, copy COMMAND.COM from your DOS diskette to that directory.

```
Insert disk with batch file
and strike any key when ready
```

INFORMATIONAL: DOS is attempting to execute the next command from a batch file, but the diskette holding the batch file was removed from the drive. This message occurs with DOS V3.1. With DOS V3.0, a fatal error is given when the diskette is changed.

Insert the diskette holding the batch file into the disk drive and press a key to continue.

```
Insert diskette for drive d and strike
any key when ready
```

INFORMATIONAL: On a system with one floppy disk drive, you or one of your programs specified the tandem disk drive `d` (A: or B:). This drive is different from the current disk drive.

If the correct diskette is in the disk drive, press a key. If not, insert the correct diskette into the floppy disk drive and then press a key.

```
Insufficient disk space
```

WARNING or ERROR: The disk does not have enough free space to hold the file being written. All DOS programs terminate when this problem occurs, but some non-DOS programs continue.

If you think the disk has enough room to hold this file, run CHKDSK to see whether the disk or diskette has a problem. Sometimes when you terminate programs early by pressing Ctrl-Break, DOS is not allowed to do the necessary clean-up work. When this happens, disk space is temporarily trapped. CHKDSK can free these areas.

If you simply have run out of disk space, free some disk space or use a different diskette or hard disk. Then try the command again.

`Insufficient memory`

ERROR: The computer does not have enough free RAM to execute the program or command.

If you loaded a resident program like PRINT, GRAPHICS, SideKick, or ProKey, restart DOS and try the command before loading any resident program. If this method fails, remove any unneeded device driver or RAM-disk software from the CONFIG.SYS file and restart DOS. If this action fails, your computer does not have enough memory for this command. You must increase your RAM to run the command.

`Intermediate file error during pipe`

ERROR: DOS is unable to create or write to one or both of the intermediate files it uses when piping (|) information between programs. The disk or root directory is full, or DOS cannot locate the files. The most frequent cause is running out of disk space.

Run the DIR command on the root directory of the current disk drive. Make sure that you have enough free space and enough room in the root directory for two additional files. If you do not, free space on the disk by deleting, or copying and deleting, files. You can also copy the necessary files to a different diskette with sufficient space.

One possibility is that a program is deleting files, including the temporary files DOS uses. If this is the case, you should correct the program, contact the dealer or program publisher, or avoid using the program with piping.

`Internal stack overflow`
`System halted`

ERROR: Your programs and DOS have exhausted the *stack*, the memory space reserved for temporary use. This problem is usually caused by a rapid succession of hardware devices demanding attention (interrupts). DOS stops, and the system must be turned off, then restarted.

The circumstances that cause this message are infrequent and erratic, and may not recur. If you want to prevent this error from occurring at all, add the STACKS directive to your CONFIG.SYS file. If the directive is already in your CONFIG.SYS file, then increase the number of stacks specified. See Chapter 14 for more information on the STACKS directive.

`Invalid COMMAND.COM in drive d`

WARNING: DOS tried to reload COMMAND.COM from the disk in drive `d` and found that the file was of a different version of DOS. You will see a message instructing you to insert a diskette with the correct version and then to press a key. Follow the directions for that message.

If you frequently use the diskette that was originally in the disk drive, copy the correct version of COMMAND.COM to that diskette.

`Invalid COUNTRY code or code page`

WARNING (start-up): Either the COUNTRY code number or the code page number given to the COUNTRY directive in the CONFIG.SYS file is incorrect or incompatible. DOS ignores the COUNTRY directive and continues to start.

Check the COUNTRY directive in your CONFIG.SYS file. Refer to Chapter 15 and ensure that the correct and compatible country code and code page numbers are specified. If you detect an error, edit and save the file and restart DOS.

`Invalid COMMAND.COM, system halted`

ERROR: DOS could not find COMMAND.COM on the hard disk. DOS halts and must be restarted.

COMMAND.COM may have been erased or the `COMSPEC=` setting in the environment may have been changed. Restart the computer from the hard disk. If you see a message indicating that COMMAND.COM is missing, that file was erased. Restart DOS from a diskette and recopy COMMAND.COM to the root directory of the hard disk or to wherever your SHELL command directs, if you have used this command in your CONFIG.SYS file.

If you restart DOS and this message appears later, a program or batch file is erasing COMMAND.COM or is altering the `COMSPEC=` parameter. If a batch file is erasing COMMAND.COM, edit the batch file. If a program is erasing COMMAND.COM, contact the dealer or publisher who sold you the program. If `COMSPEC=` is being altered, either edit the offending batch file or program, or place COMMAND.COM in the subdirectory your program or batch file expects.

`Invalid directory`

ERROR: One of the following errors occurred: (1) you specified a nonexistent directory name; (2) you misspelled the directory name; (3) the directory path is on a different disk; (4) you forgot to give the path character (\) at the beginning of the name; or (5) you did not separate the directory names with the path character. Check your directory names, ensure that the directories do exist, and try the command again.

`Invalid disk change`

WARNING: The diskette in the 720K, 1.2M, or 1.44M disk drive was changed while a program had open files to be written to the diskette. You will see the message `Abort, Retry, Fail`. Place the correct diskette in the disk drive and choose **R** for Retry.

`Invalid drive in search path`

WARNING: One specification you gave to the PATH command has an invalid disk drive name, or a named disk drive is nonexistent or hidden temporarily by a SUBST or JOIN command.

Use PATH to check the paths you instructed DOS to search. If you gave a nonexistent disk drive name, use the PATH command again and enter the correct search paths. If the problem is temporary because of a SUBST or JOIN command, you can use PATH again to enter the paths, but leave out or correct the wrong entry. Or you can just ignore the warning message.

`Invalid drive specification`

ERROR: This message is given when one of the following errors occurs: (1) you entered the name of an invalid or nonexistent disk drive as a parameter to a command; (2) you gave the same disk drive for both source and destination, which is not permitted for the command; or (3) by not giving a parameter, you defaulted to the same source and destination disk drive.

Remember that certain DOS commands, such as SUBST and JOIN, temporarily hide disk drive names while the command is in effect. Check the disk drive names. If the command is objecting to a missing parameter and defaulting to the wrong disk drive, explicitly name the correct disk drive.

```
Invalid drive specification
Specified drive does not exist,
or is non-removable
```

ERROR: One of the following errors occurred: (1) you gave the name of a nonexistent disk drive; (2) you named the hard disk drive when using commands for diskettes only; (3) you did not give a disk drive name and defaulted to the hard disk when using commands for diskettes only; or (4) you named or defaulted to a RAM-disk drive when using commands only for a "real" floppy diskette.

Remember that certain DOS commands, such as SUBST and JOIN, temporarily hide disk drive names while the command is in effect. Check the disk drive name you gave and try the command again.

`Invalid environment size specified`

WARNING: You gave the SHELL directive in CONFIG.SYS. The environment-size switch */E:size* contains either nonnumeric characters or a number that is less than 160 or greater than 32768.

If you are using the SHELL */E:size* switch of DOS V3.1, *size* is the number of 16-byte memory blocks, not the number of bytes.

Check the form of your CONFIG.SYS SHELL directive; the form needs to be exact. There must be a colon between */E* and *size*; there must not be a comma or space either between or within the */E:* and the *size* characters; and the number in *size* should be greater than or equal to 160, but less than or equal to 32768.

`Invalid number of parameters`

ERROR: You gave either too few or too many parameters to a command. One of the following errors occurred: (1) you omitted required information; (2) you forgot a colon immediately after the disk drive name; (3) you put a space in the wrong place or omitted a needed space; or (4) you forgot to place a slash (/) in front of a switch.

`Invalid parameter`
`Incorrect parameter`

ERROR: At least one parameter you entered for the command is not valid. One of the following occurred: (1) you omitted required information; (2) you forgot a colon immediately after the disk drive name; (3) you put a space in the wrong place or omitted a needed space; (4) you forgot to place a slash (/) in front of a switch; or (5) you used a switch not recognizable to the command. Also check this message under the individual command for more information.

`Invalid partition table`

ERROR (start-up): While you were attempting to start DOS from the hard disk, DOS detected a problem in the hard disk's partition information.

Restart DOS from a diskette. Back up all files from the hard disk if possible. Run FDISK to correct the problem. If you change the partition information, you must reformat the hard disk and restore all its files. Refer to Chapter 5 for more information on using FDISK.

`Invalid path`

ERROR: One of the following errors has occurred to a path name you entered: (1) the path name contains illegal characters; (2) the name has more than 63 characters; or (3) one of the directory names within the path is misspelled or does not exist.

Check the spelling of the path name. If needed, issue the DIR command for that disk and ensure that the directory you specified does exist and that you have the correct path name. Be sure that the path name contains 63 characters or less. If necessary, change the current directory to one "closer" to the file and shorten the path name.

`Invalid STACK parameter`

WARNING (start-up): One of the following errors occurred to the STACKS directive in your CONFIG.SYS file: (1) a comma is missing between the number of stacks and the size of the stack; (2) the number of stack frames is not in the range of 8 to 64; (3) the stack size is not in the range of 32 to 512; (4) you omitted either the number of stack frames or the stack size; or (5) either the stack frame or the stack size (but not both) is 0. DOS continues to start but ignores the STACKS directive.

Check the STACKS directive in your CONFIG.SYS file. Edit and save the file, then restart DOS.

`Invalid switch character`

WARNING: You have used RAMDISK.SYS in your CONFIG.SYS file. RAMDISK encountered a switch (/) but the character immediately following it was not an *E* for *extended memory*. DOS loads RAMDISK and attempts to install RAMDISK in low (nonextended) memory. Either you misspelled the /E switch, or you left a space between the / and the E. Edit and save your CONFIG.SYS file, then restart DOS.

`Memory allocation error`
`Cannot load COMMAND, system halted`

ERROR: A program destroyed the area where DOS keeps track of in-use and available memory. You must restart DOS.

If this error occurs again with the same program, the program may have a flaw or simply does not work well with other programs. Use a backup copy of the program. If the problem persists, contact the dealer or program publisher.

`Missing operating system`

ERROR (start-up): The DOS hard disk partition entry is marked as "bootable" (able to start DOS), but the DOS partition does not have a copy of DOS on it. DOS does not boot.

Start DOS from a diskette. If you have existing files on the hard disk, back up the files. Issue **FORMAT d: /S** to put a copy of the operating system on the hard disk. If necessary, restore the files that you backed up.

`No free file handles`
`Cannot start COMMAND, exiting`

ERROR: DOS could not load an additional copy of COMMAND.COM because no file handles (`FILES=`) were available.

Using the FILES command, edit the CONFIG.SYS file on your start-up disk to increase the number of file handles by five. Restart DOS and try the command again.

```
Non-System disk or disk error
Replace and strike any key when ready
```

ERROR (start-up): Your diskette or hard disk does not contain DOS, or a read error occurred when you started the system. DOS does not boot.

If you are using a floppy disk system, insert a bootable diskette in drive A: and press a key.

The most frequent cause of this message on hard disk systems is that you left a nonbootable diskette in disk drive A: with the door closed. Open the door to disk drive A: and press a key. DOS will boot from the hard disk.

```
Not enough memory
```

ERROR: The computer does not have enough free RAM to execute the program or command.

If you loaded a memory-resident program such as PRINT, GRAPHICS, SideKick, or ProKey, restart DOS and try the command again before loading any resident program. If this method fails, remove any unneeded device driver or RAM-disk software from the CONFIG.SYS file and restart DOS. If this option fails also, your computer probably does not have enough memory for this command. Run CHKDSK to determine how much memory you have left. You may need to increase your RAM to run the command.

```
Out of environment space
```

WARNING: DOS is unable to add to the environment any more strings from the SET command. The environment cannot be expanded. This error occurs when you load a resident program, such as MODE, PRINT, GRAPHICS, SideKick, or ProKey.

If you are running DOS V3.1 or later, refer to the SHELL command in Chapter 16 for information about expanding the default space for the environment. DOS V3.0 has no method to expand the environment.

```
Path not found
```

ERROR: A file or directory path you named does not exist. You may have misspelled the file name or directory name, or you omitted a path character (\) between directory names or between the final directory name and file name. Another possibility is that the file or directory does not exist where you specified. Check these possibilities and try again.

```
Path too long
```

ERROR: You gave a path name that exceeds the 63-character limit of DOS. Either the name is too long, or you omitted a space between file names. Check the command line. If the phrasing is correct, you must change to a directory closer to the file you want and try the command again.

`Program too big to fit in memory`

ERROR: The computer does not have enough memory to load the program or command you invoked.

If you have any resident programs loaded (such as PRINT, GRAPHICS, or SideKick), restart DOS and try the command again without loading the resident programs. If this message appears again, reduce the number of buffers (`BUFFERS=`) in the CONFIG.SYS file; eliminate unneeded device drivers or RAM-disk software; and restart DOS. If these actions do not solve the problem, your computer may in fact lack the RAM for the program or command. Run CHKDSK to determine the memory available. Or you may want to consult your software dealer for advice about memory requirements for these programs on you computer.

`Sector size too large in file filename`

WARNING: The device driver is inconsistent. The device driver defined a particular sector size to DOS, but attempted to use a different size. Either the copy of the device driver is bad, or the device driver is incorrect. Copy a backup of the device driver to the boot diskette and then reboot DOS. If the message appears again, the device driver is incorrect. If you wrote the driver, correct the error. If you purchased the program, contact the dealer or software publisher.

`Sharing violation`

WARNING: With the file-sharing program (SHARE.EXE) loaded, you or one of your programs attempted to access a file by using a sharing mode not allowed at this time. Another program or computer has temporary control over the file.

You will see the message `Abort, Retry, Ignore`. Choose **R** for Retry several times. If the problem persists, choose **A** for Abort. If you abort, however, any data currently being manipulated by the program is lost.

`Syntax error`

ERROR: You phrased a command improperly by (1) omitting needed information, (2) giving extraneous information, (3) leaving an extra space in a file name or path name, or (4) using an incorrect switch. Check the command line for these possibilities and try the command again.

`Too many block devices`

WARNING (start-up): There are too many DEVICE directives in your CONFIG.SYS file. DOS continues to start, but does not install any additional device drivers.

DOS can handle only 26 block devices. The block devices created by the DEVICE directives plus the number of block devices automatically created by DOS exceed this number. Remove any unnecessary DEVICE directives in your CONFIG.SYS file and restart DOS.

`Top level process aborted, cannot continue`

ERROR (start-up): COMMAND.COM or another DOS command detected a disk error, and you chose the **A** (Abort) option. DOS cannot finish starting itself, and the system halts.

Try to start DOS again. If the error recurs, use a floppy diskette (if starting from the hard disk) or another floppy diskette (if starting from floppy diskettes) to start DOS. After it has started, use the SYS command to put another copy of the operating system on the disk, and then copy COMMAND.COM to the disk. If DOS reports an error during the copying, the disk or diskette is bad. Either reformat or retire the floppy diskette, or back up and reformat the hard disk.

`Unable to create directory`

ERROR: Either you or a program attempted to create a directory, and one of the following occurred: (1) a directory by the same name already exists; (2) a file by the same name already exists; (3) you are adding a directory to the root directory and the root directory is full; or (4) the directory name has illegal characters or is a device name.

Issue the DIR command for the disk. Make sure that no file or directory already exists with the same name. If adding the directory to the root directory, remove or move (copy, then erase) any unneeded files or directives. Check the spelling of the directory and ensure that the command is properly phrased.

`Unrecognized command in CONFIG.SYS`

WARNING (start-up): DOS detected an improperly phrased directive in CONFIG.SYS. The directive is ignored, and DOS continues to start, but DOS does not indicate the incorrect line. Examine the CONFIG.SYS file, looking for improperly phrased, misspelled, or incorrect directives. Edit the line, save the file, and restart DOS.

DOS Device Error Messages

When DOS detects an error while reading or writing to disk drives or other devices, one of the following messages appears:

type `error reading` *device*

type `error writing` *device*

type is the type of error, and *device* is the device at fault. If the device is a floppy disk drive, do not remove the diskette from the drive. Refer to the possible causes and corrective actions described in this section, which lists the types of error messages that may appear.

`Bad call format`

A device driver was given a requested header with an incorrect length. The problem is with the applications software making the call.

`Bad command`

The device driver issued an invalid or unsupported command to the device. The problem may be with the device driver software or with other software trying to use the device driver.

`Bad format call`

The device driver at fault passed an incorrect header length to DOS. If you wrote this device driver, you must rewrite it to correct the problem. For a purchased program, contact the dealer or publisher who sold you the driver.

`Bad unit`

An invalid subunit number was passed to the device driver. The problem may be with the device driver software or with other software trying to use the device driver. Contact the dealer who sold you the device driver.

`Data`

DOS could not correctly read or write the data. Usually the disk has developed a defective area.

`Drive not ready`

An error occurred while DOS tried to read or write to the disk drive. For floppy disk drives, the drive door may be open, the diskette may not be inserted correctly, or the diskette may not be formatted. For hard disk drives, the drive may not be properly prepared or you may have a hardware problem.

`FCB unavailable`

With the file-sharing program (SHARE.EXE) loaded, a program that uses the DOS V1 method of file handling attempted to open concurrently more file control blocks than were specified with the FCBS directive.

Use the **A** (Abort) option (see the end of the section). Increase the value of the FCBS CONFIG.SYS command (usually by four) and reboot the system. If the message reappears, increase the number again and reboot.

`Lock violation`

With the file-sharing program (SHARE.EXE) or network software loaded, one of your programs attempted to access a file that is locked. Your best choice is **R** for Retry. Then try **A** for Abort. If you choose **A**, however, any data in memory is lost.

`No paper`

The printer is either out of paper or not turned on.

`Non-DOS disk`

The FAT has invalid information. This diskette is unusable to DOS. You can abort and run CHKDSK on the diskette to see whether any corrective action is possible. If CHKDSK fails, your alternative is to reformat the diskette. Reformatting, however, will destroy any remaining information on the diskette. If you use more than one operating system, the diskette may have been formatted under that system and should not be reformatted with the DOS FORMAT command. Use the other operating system to check the disk.

`Not ready`

The device is not ready and cannot receive or transmit data. Check the connections, make sure that the power is on, and check to see whether the device is ready.

`Read fault`

DOS was unable to read data, usually from a hard disk or diskette. Check the disk drive doors and be sure that the diskette is inserted properly.

`Sector not found`

The disk drive was unable to locate the sector on the diskette or hard disk platter. This error is usually the result of a defective spot on the disk or of defective drive electronics. Some copy-protection schemes also use the defective-spot method to prevent unauthorized duplication.

`Seek`

The disk drive could not locate the proper track on the diskette or hard disk platter. This error is usually the result of a defective spot on the diskette or hard disk platter, an unformatted disk, or drive electronics problems.

`Sharing violation`

With the file-sharing program (SHARE.EXE) or network software loaded, your programs attempted to access a file by using a sharing mode not specified for that file. Your best response is **R** for Retry; if that fails, try **A** for Abort.

`Write fault`

DOS could not write the data to this device. Perhaps you inserted the diskette improperly or you left the disk drive door open. Another possibility is an electronics failure in the floppy or hard disk drive. The most frequent cause is a bad spot on the diskette.

`Write protect`

The diskette is write-protected.

Note: One of the previously listed messages (usually `Data`, `Read fault`, or `Write fault`) appears when you are using a double-sided diskette in a single-

sided disk drive or a 9-sector diskette (V2 and later) with DOS V1. DOS will display one of these error messages followed by this line:

```
Abort, Retry, Ignore?
```

If you choose **A** for Abort, DOS ends the program that requested the read or write condition. Choosing **R** for Retry causes DOS to try the operation again. If you choose **I** for Ignore, DOS skips the operation, and the program continues. However, some data may be lost when **I** is used.

The order of preference, unless stated differently under the message, is **R**, **A**, and **I**. You should retry the operation at least twice. If the condition persists, you must decide whether to abort the program or ignore the error. If you ignore the error, data may be lost. If you abort, data still being processed by the program and not yet written to the disk will be lost. Remember that **I** is the least desirable option and that **A** should be used after **R** has failed at least two times.

APPEND *V3.3, V4—External*

Instructs DOS to search the specified directories on the specified disks for non-program/nonbatch files.

Syntax

To establish the data-file search path the first time, use the following form:

*dc:pathc***APPEND** *d1:path1;d2:path2;d3:path3;...*

dc: is the name of the disk drive that holds the command.

pathc\ is the path to the command.

d1:, *d2:*, *d3:* are valid disk drive names.

path1, *path2*, *path3* are valid paths to the directories you want DOS to search for nonprogram/nonbatch files. The three periods (...) represent additional disk drive and path names.

To use either of the APPEND switches, use the following:

*dc:pathc***APPEND** */X /E*

DOS V4 users can also use the following syntax for the */X* switch:

*dc:pathc***APPEND /X:OFF**dc:pathc***APPEND /X:ON**

To change the data-file search path, use the following:

APPEND *d1:path1,d2:path2;d3:path3;...*

With DOS V4, the default state for a search for files that have the drive or path specified is as follows:

APPEND */PATH:ON*

To turn off a search for files that have the drive or path specified (for DOS V4 only), use the following form:

APPEND */PATH:OFF*

To see the search path, use the following form:

APPEND

To disconnect the data-file search, use the following form:

APPEND;

Switches

/X	Redirects programs that use the DOS function calls SEARCH FIRST, FIND FIRST, and EXEC.
/X:ON	Same as /X (DOS V4 only).
/X:OFF	Turns this feature off (DOS V4 only).
/E	Places the disk drive paths in the environment.
/PATH:ON	Turns on search for files that have drive or path specified (DOS V4 only).
/PATH:OFF	Turns off search for files that have drive or path specified (DOS V4 only).

Reference

See Chapter 6.

Rules

1. The first time you execute APPEND, the program loads from the disk and installs itself in DOS. APPEND then becomes an internal command and is not reloaded from the disk until you restart DOS.
2. You can give the /X and /E switches only when you first invoke APPEND. You cannot give any path names with these two switches.
3. If you do not enter a disk drive name for a path, APPEND uses the current disk drive.
4. Each path can be any valid series of subdirectories, separated by the path character \ (the backslash).
5. If you specify more than one set of paths, the following rules apply:
    a. The path sets must be separated by semicolons.
    b. The search for the nonprogram files is made in the order in which you gave the path sets. First the specified directory (or current

directory if none is specified) is searched. Then *d1:path1* is searched, followed by *d2:path2*, and so on, either until the file is found or when APPEND exhausts the list of directory paths.

6. The length of the paths given for APPEND cannot exceed 127 characters, which includes the APPEND command itself.
7. If an invalid path is encountered, such as a misspelled path or one that no longer exists, DOS skips the path and does not display any message.
8. If you use the /X switch, APPEND processes the DOS function calls SEARCH FIRST, FIND FIRST, and EXEC.
9. With DOS V4, you can disable the /X switch using **/X:OFF**.
10. Do not use RESTORE while you are using the /X switch.
11. To disable APPEND, give the command followed by a semicolon. APPEND remains resident but inactive until you issue another APPEND command with path names.
12. If you use the /E switch, DOS establishes an environmental variable named APPEND. The variable holds the current paths that APPEND uses.
13. The file found by APPEND can be safely read by the program. However, any changes to the file are saved in a copy of the file placed in the current directory. The original file remains intact.
14. If you are using the ASSIGN command with APPEND, you must issue the APPEND command before you use ASSIGN.

Examples

1. **APPEND /X /E**

   Loads the APPEND command and establishes APPEND to intercept the SEARCH FIRST, FIND FIRST, and EXEC DOS function calls (the /X switch). DOS places in the environment the paths you give to APPEND (the /E switch). This example assumes that APPEND is in the current directory or on the PATH.

2. **APPEND C:\BIN;C:\BIN\OVR**

   Instructs APPEND to search the specified or current directory for nonprogram files and then search the directories C:\BIN and C:\BIN\OVR.

3. **APPEND /PATH:ON**

   This example instructs APPEND to ignore invalid drive or path designations and search the APPEND path for the file. If you enter the command **TYPE C:\TEMP\DOCUMENT.TXT**, but the directory C:\TEMP does not exist, the APPEND paths then will be searched for DOCUMENT.TXT.

4. **APPEND;**

   Disables the APPEND command. The APPEND command stays in memory until you restart DOS. To reactivate APPEND, simply give a set of path names to the command.

Notes

APPEND is the counterpart of the PATH command, which works with program files. APPEND is the PATH command for data and other nonprogram files. The command is especially helpful for users whose programs do not support hierarchical directories. With the APPEND command, you can place additional program files (such as those used with WordStar Release 3) or data files in any directory.

You can use APPEND /X to trick programs at three levels. First, when a program attempts to open a file to read, APPEND enables the program to find the file to be opened. Second, when a program simply searches for a file name, APPEND enables the program to find the file name. Third, when a program attempts to execute another program, APPEND helps to find the program and then executes it.

The /E switch places the paths fed to APPEND in the environment, under the variable named APPEND. If you use the /E switch, programs can examine the environment for the APPEND variable and use the contents to find their files. However, do not use APPEND from within a program if you have used the /E switch. The changes that APPEND makes to the environment are temporary—the copy of COMMAND.COM used to run APPEND is temporary. When you return to the program, changes made by APPEND will be lost.

The command APPEND /X can cause problems with certain programs. If you suddenly encounter a problem with locating files, first deactivate APPEND, using the command **APPEND;**. If the problem persists, use **APPEND** with the /X switch disabled. If the problem continues, look for some other cause. If APPEND is deactivated, it should not affect any running program.

Do not use APPEND with RESTORE. RESTORE will search for files in the directories on which you have used APPEND. If you are using the /N or /M switch with RESTORE, the correct files may not be processed. Before you use RESTORE, be sure that you have deactivated APPEND.

Be cautious if you are writing to files that APPEND has found. APPEND tricks programs into reading files from any location. APPEND does not trick programs when they write files. DOS saves any changes you make to a file in a copy of that file, which it places in the current directory. The original file in the directory on which you used APPEND remains unchanged. This file duplication can be confusing when you use a file in one directory and save the new version, and then move to another directory and attempt to reuse the file. You will be working with the original file without the changes, while the altered file remains in the directory from which you made the changes.

APPEND is an "open-me-first" command if you use it with the ASSIGN command. You must start APPEND before you use the ASSIGN command. Reversing the order causes APPEND to deliver an error message, and the command does not load.

Messages

1. `APPEND/ASSIGN Conflict`
   `APPEND/TopView Conflict`

   ERROR: You attempted to use APPEND after you loaded ASSIGN, TopView or DESQview. If you are using TopView or DESQview, exit the program, give the APPEND command, and then restart your program. If you are using ASSIGN, break the assignment, issue the APPEND command, and then issue the ASSIGN command again.

2. `APPEND already installed`

   WARNING: You attempted to load APPEND again by giving a disk drive or path name for a command. You need to load APPEND only once. Give the command again without the command's drive or path name.

3. `Incorrect APPEND Version`

   ERROR: You are using a version of APPEND from a different version of DOS. Make certain that you are not using the version of APPEND from the IBM Local Area Network (LAN) program. The problem may be that the wrong version of APPEND is being loaded first from a PATHed directory.

4. `Invalid parameter`

   ERROR: You used a switch that APPEND did not recognize when you first ran the command. The only valid switches for APPEND are /X and /E. Try the command again using the correct switches.

5. `Invalid Path`

   ERROR: The path you gave to APPEND was not valid, for one of these reasons: (1) you used illegal characters, (2) you forgot to separate multiple path sets with semicolons, or (3) you gave a path name that exceeded 63 characters. Check the spelling of each path set, the semicolons that separate multiple paths, and the length of each path. Then try the command again.

6. `Invalid path or parameter`

   ERROR: After you installed APPEND, you made one of the following errors: (1) you used a switch with the command, (2) you misspelled a path name, or (3) you used a path that exceeded 63 characters. Check the spelling of the paths and the semicolons that separate multiple paths. Then try the command again. To use a different switch with APPEND, you must restart DOS and give the APPEND command with the appropriate switch.

7. `No Append`

   INFORMATION: You typed APPEND to see the current path, and APPEND is currently inactive.

ASSIGN

V2, V3, V4—External

Instructs DOS to use a disk drive other than the one specified by a program or command.

Syntax

*dc:pathc***ASSIGN d1=d2...**

dc: is the disk drive holding the command.

pathc\ is the path to the command.

d1 is the letter of the disk drive the program or DOS normally uses.

d2 is the letter of the disk drive that you want the program or DOS to use instead of the usual drive.

The three periods (**...**) represent additional disk drive assignments.

To clear the reassignment, use the following form:

*dc:pathc***ASSIGN**

Reference

See Chapter 9.

Rules

1. You can reassign any valid DOS drive to any other drive, but not to itself.
2. Do not use a colon after the disk drive letter for **d1** or **d2**. You can use a space on either side of the equal sign.
3. You can give more than one assignment on the same line. Use a space between each set of assignments, as in the following example:

   ASSIGN B=C A=C

4. Use ASSIGN only when necessary.
5. Do not ASSIGN a disk drive to a drive that is not on your system. If you make this mistake, DOS will return the error message:

   ```
   Invalid parameter
   ```

Examples

1. **ASSIGN A = C** or **ASSIGN A=C**

   DOS reroutes to drive C: any activity for drive A:. You can use a space on either side of the equal sign.

2. **ASSIGN A=C B=C**

   DOS reroutes to drive C: any activity for drives A: and B:.

3. **ASSIGN**

   Clears any previous drive reassignment.

Notes

ASSIGN is a pretender command for DOS. ASSIGN reroutes a program, causing it to use a disk drive that is different from the one the program intends. A program "thinks" that it is using a certain disk drive, when in fact the program is using another.

You should use ASSIGN primarily for programs that do not allow you to specify the disk drive. Some accounting and word processing programs assume only two possible disk drives—the minifloppy disk drives A: and B:. You use ASSIGN to trick programs into using the hard disk (drive C:). Computer users need to use this command less frequently now, because most programs allow users to specify drives exceeding the former limitation of A: and B:.

The ASSIGN command does not work with all programs. For example, you cannot use this command with DISKCOMP, DISKCOPY, and FORMAT. Some programs make a special DOS request to read a specific part of the diskette in the floppy disk drive. ASSIGN cannot intercept and reroute these special requests. In such cases, your only choice is to place the diskette requested by the program in the correct disk drive.

Do not ASSIGN a real disk drive to a nonexistent drive. If you attempt such an assignment, the DOS V3 and V4 ASSIGN command immediately displays the following error message:

```
Invalid parameter
```

If you have not used the LASTDRIVE command in CONFIG.SYS, when you type the line:

ASSIGN A=F

DOS immediately displays an `Invalid parameter` error message and does not change any current assignments.

With DOS V3.1 and later versions, an alternative to ASSIGN is SUBST, which assigns a disk drive letter to a subdirectory (see the SUBST command).

ASSIGN can interfere with certain DOS commands, such as DIR, CHKDSK, DISKCOMP, DISKCOPY, BACKUP, and RESTORE.

When ASSIGN is in effect, these DOS commands will not work on the drive on which you have used ASSIGN. You must break the assignment (by issuing ASSIGN with no parameters) before you can perform these commands on the disk.

Message

`Invalid parameter`

ERROR: You made one of the following errors: (1) you put a colon after one or more drive letters, (2) you did not type all the information ASSIGN wanted, (3) you used a letter for a nonexistent disk drive, or (4) you did not correctly phrase the command.

ATTRIB

V3, V4—External

Displays, sets, or clears a file's read-only or archive attributes.

Syntax

To set the file's attributes on, use the following form:

*dc:pathc***ATTRIB** *+R +A d:path***filename.ext** */s*

To clear the file's attributes, use the following form:

*dc:pathc***ATTRIB** *−R −A d:path***filename.ext** */s*

To display a file's read-only and archive status, use the following form:

*dc:pathc***ATTRIB** *d:path***filename.ext** */s*

dc: is the name of the disk drive holding the command.

pathc\ is the path to the command.

+R sets a file's read-only attribute on.

+A sets a file's archive attribute on.

−R sets a file's read-only attribute off.

−A sets a file's archive attribute off.

d: is the name of the disk drive holding the files for which the read-only attribute will be displayed or changed.

path\ is the path to the files for which the read-only attribute will be displayed or changed.

filename.ext is the name of the file(s) for which the read-only attribute will be displayed or changed. Wild cards are permitted.

Switch

/S Sets or clears the attributes of the specified files in the specified directory and all subdirectories to that directory.

Reference

See Chapter 7.

Rules

1. If you do not give a disk drive name (*d:*), ATTRIB uses the current disk drive.
2. If you do not give a path name (*path*\), ATTRIB uses the current directory.
3. If you do not give a file name (**filename.ext**), DOS displays the error message:

```
Invalid path or filename not found
```

If you give an incorrect file name, DOS displays the error message:

```
Invalid parameters
```

4. To mark a file as read-only, insert **+R** (for *add read-only*) between the command and the file name. To remove the read-only flag from the file, insert **−R** (for *subtract read-only*) between the command and the file name.

5. To mark a file's archive attribute, insert **+A** (for *archive on*) between the command and the file name. To clear the archive attribute from the file, insert **−A** (for *archive off* between the command and file name or after the file name.

6. To display the read-only and archive-attribute status of a file, do not use **+R**, **−R**, **+A**, or **−A**. ATTRIB displays an `A` in the third column before the file names if the archive attribute is on. An `R` is displayed in the seventh column if the read-only attribute is on for the file. If the attribute is not on, a space appears in place of the letter.

7. You can use both the **R** and the **A** characters, or either character, in the command. You can mix the plus and minus attributes of the commands (for example, **+R −A**), but you cannot use plus and minus the same character in the command (for example, **+R −R**).

8. When a file is marked as read-only, you cannot alter or delete it.

Notes

With the DOS V3 addition of ATTRIB, users have more control over the DOS read-only and archive file attributes. ATTRIB sets or clears these attributes. When the file is marked as read-only, it cannot be altered or deleted. When the archive attribute is on, the file can be processed by using the commands **BACKUP /M**, **XCOPY /M**, or **XCOPY /A**.

ATTRIB does not affect any other DOS file attributes, including those for system and hidden files. To change these file characteristics, you must use a different program, such as the Norton Utilities or SDACHMD (by SDA Associates).

Although you cannot alter or erase a read-only file, you can rename the file. However, if you change the name of a read-only file, you can cause some problems with certain programs.

Messages

1. `Access denied`

   ERROR: You cannot change the read-only attribute of this file. The file may be temporarily unavailable because another computer or program is using the file, or the file may be in a directory on a networked disk for which you do not have permission to change files. If another program or computer has control of the file, try the command again later. If you do not have permission to change the file on the networked disk, ask the user of the other disk to make the change for you.

2. `Invalid number of parameters`

   ERROR: You did not specify any file names.

3. `Invalid path or file not found`

   ERROR: You specified a directory or file name that does not exist. Check the spelling of the names and be sure that the files are in the directory you specified.

4. `Syntax error`

   ERROR: One of the following errors occurred: (1) you gave an invalid switch to ATTRIB, (2) you gave an ill-formed switch command (omitting the + or −, omitting the A or R, or giving both the + and −, (3) you gave more than one file name.

BACKUP *V2, V3, V4—External*

Backs up one or more files from a hard disk or floppy disk onto another disk.

Syntax

*dc:pathc***BACKUP d1:***path****filename.ext* **d2:** */S /M /A /D:date /T:time /F /L:dl:filenamel:extl*

dc: is the disk drive that holds the command.

pathc\ is the path to the command.

d1: is the hard disk or floppy disk drive to be backed up.

path\ is the starting directory path for backup.

filename.ext specifies the name(s) of the file(s) you want to back up. Wild cards are allowed.

d2: is the hard disk or floppy disk drive that receives the backup files.

Switches

/S Backs up all *subdirectories*, starting with the specified or current directory on the source disk and working downward.

/M Backs up all files *modified* since the last time they were backed up.

/A *Adds* the file(s) to be backed up to the files already on the specified floppy disk drive.

/D:date Backs up any file that was created or changed on or after the specified *date*.

/T:time Backs up any file that was created or changed on or after the specified *time* on the specified date (given with the */D* switch).

/F *Formats* the destination floppy disk if the floppy disk is not formatted. DOS V4 will format the target floppy disks automatically without the use of this switch.

/F:size With DOS V4, *formats* the destination floppy disk according to the *size* specified. If you have a 1.2M disk drive, but only 360K floppy disks, you can specify **/F:360** to format the 360K disk in the 1.2M drive.

/L:dl:path1\filename1.ext Creates a *log* file.

Exit Codes

0	Successful backup
1	No files found to be backed up
2	Some files not backed up because of sharing problems
3	Aborted by the user (Ctrl-Break or Ctrl-C)
4	Aborted because of an error

Reference

See Chapter 7.

Rules

1. You must give both a source and a destination for BACKUP. Neither disk can be a networked disk, and neither disk can be used in an ASSIGN, SUBST, or JOIN command.
2. The source name must specify the valid name of the disk drive to be backed up. The source name also can include either or both of the following:
   a. A valid directory path containing the files to be backed up.
   b. A file name with appropriate extensions, if desired. Wild cards are allowed.
3. The destination is any valid name for a floppy or hard disk drive.
4. If you fail to give a source drive name, the error message `No source drive specified` is displayed.
5. If you do not give a source directory path, the current directory is used.
6. If you do not give a source file name, all files in the specified directory are backed up.
7. If you fail to give the drive, directory, or file name for the source, a `No source drive specified` error message is displayed.

8. If you are backing up onto diskettes with BACKUP from a version of DOS prior to V3.3, you must have a sufficient number of formatted diskettes before you invoke the BACKUP command. BACKUP does not work on unformatted or non-DOS diskettes. If you have too few formatted diskettes, you will need to stop and restart BACKUP.

   If you are using diskettes with BACKUP, with a DOS version V3.3 or after, you can give the /F switch, which causes BACKUP to run FORMAT for any unformatted diskettes. If you run out of formatted diskettes and have not given the /F switch, you will have to restart BACKUP after you stop to format more diskettes. Diskettes will be formatted to the capacity of the drive.

   If you are using BACKUP V4, you can specify the capacity of the diskette by using **/F:size**, with **size** designating diskette capacity. For example, to format a 360K diskette in a 1.2M drive, use **/F:360**.

9. If you are backing up onto a hard disk, the disk needs sufficient room to hold the backup files. If you run out of space on the hard disk, you will need to delete files from the backed-up hard disk and start BACKUP again.

10. To keep the files on a previously backed-up disk, use the **/A** option. If you do not use this switch, all files previously stored on the receiving diskette are destroyed, or all files in the \BACKUP subdirectory of the hard disk are destroyed.

    To use the /A option with diskettes, you must start with a diskette that contains the special files created by BACKUP, named `CONTROL.xxx` and `BACKUP.xxx` (`xxx` is the number of the diskette). Otherwise, BACKUP displays the message `Last backup diskette not inserted` and aborts.

11. If you do not use the /A option and are backing up onto diskettes, DOS prompts you to insert the first backup diskette. If you do use the /A option, the backup diskette should be in the correct drive before you type the command line; otherwise, DOS requests that you `Insert last backup diskette`.

12. When you are backing up with diskettes, DOS prompts you to change diskettes. Make sure that you insert the correct diskette in the disk drive when you are prompted.

13. You can retrieve backed-up files only with the RESTORE command. Otherwise, these files are unusable.

14. BACKUP follows the VERIFY command. If VERIFY is OFF, DOS does not check the newly created BACKUP files on the diskette to see whether they were recorded correctly. If VERIFY is ON, DOS checks to ensure that the files were stored correctly.

15. If you are backing up onto diskettes, all files are placed in the root directory. If you are backing up onto another hard disk, the files are stored in the subdirectory \BACKUP.

16. To create a log of the files that BACKUP processed, give the /**L** switch. If you do not specify a full file name, BACKUP creates the log file under the name `BACKUP.LOG` in the source disk's root directory.

17. If you do not specify a disk drive (*dl:*) for the log file name, the source disk is used.

18. If you do not specify a directory path (*pathl*) for the log file name, the current directory is used.

19. If you do not specify a file name (*filenamel.extl*) for the log file, BACKUP displays an error message and aborts.

    If you give a log file name, you must use a colon between the L and the first character in the name. Do not use spaces from the beginning of the switch to the end of the log file name. If you do not give a log file name, do not use a colon after the L.

    If a log file already exists, BACKUP will add to the log file. If a file does not exist, a new file is created.

    You cannot place the log file on the destination disk (the disk used to store the backup files).

Examples

For the following examples, assume that the hard disk is C: and that the following subdirectories exist:

```
C:\
C:\DOS
C:\WP
C:\DATA
C:\DATA\LETTERS
C:\DATA\MEMOS
```

1. Backing Up the Entire Hard Disk

   BACKUP C:\ A: /S

   Here, the source to back up is specified as **C:**. The **/S** switch instructs BACKUP to backup all subdirectories of the source—that is, all subdirectories of the root directory.

2. Backing Up a Specific Subdirectory

   BACKUP C:\DATA\LETTERS A:

   This example shows how to back up only a single subdirectory. In this case, the subdirectory **LETTERS** will be backed up. You do not need to use any switches.

   A common mistake occurs when you are working in a subdirectory and you try to use D-1 or E-2 to back up the entire disk. To back up the entire disk, use **D-2 or E-1**.

3. Backing Up Several Subdirectories

   BACKUP C:\DATA A: /S

   In this example, **\DATA** is the source to back up. Because of the **/S** switch, all subdirectories of \DATA are backed up.

4. Backing Up All Files as of a Specified Date

   BACKUP C:\ A: /S /D:08/21/89

   In this example, all files that were created or modified on or after August 21, 1989, are backed up. Note that the root directory is the source and that the /S switch is given to back up all subdirectories of the source directory.

5. Backing Up Modified Files Only

   BACKUP C:\ A: /S /M

   The /M switch tells BACKUP to back up any files that have the archive bit set on. When you create or modify a file, the archive bit is set on. When you back up a file, BACKUP turns off the archive bit.

   When you use the /M switch, BACKUP skips any file with an archive bit that is turned off. This switch setting allows BACKUP to back up any file or version of a file you have not previously backed up or changed since the last backup.

   If you select this method of backing up, do not use a previous set of backup disks for this operation. All files on the disks are erased before this disk is used for backup. If a file that was backed up on the earlier set of disks has not been modified since the last backup, it will not be backed up this time. Because the backup disks are erased by BACKUP prior to use, the backup copy of the file will be lost.

6. Adding Modified Files to the Current Backup Disks

   BACKUP C:\ A: /S /M /A

   As with the previous example, only modified files are backed up. But, in this case, BACKUP does not overwrite files on the backup disk. Instead, you see the message

   ```
   Insert last backup diskette in drive A:
   Press any key to continue ...
   ```

   BACKUP begins to add modified files to the backup disks, because of the /A switch.

   When you give the /F switch with BACKUP to run FORMAT, BACKUP tests each disk to see if it has been formatted. If not, BACKUP calls the FORMAT program. DOS does not give the customary message to place a diskette in the drive. BACKUP and FORMAT assume that the diskette you have placed in the drive is the correct diskette. When FORMAT runs, you can format as many diskettes as you wish; remember, however, to change diskettes before you format another.

If you are using DOS V3.3, the disk you place in the drive must be of the same capacity as the drive. You cannot, for example, place a 720K diskette in a 1.44M drive. Users of DOS V4.0 or V4.01 do not have this restriction. You may give the disk size with the /F switch. For example, use **/F:720** to format a disk.

Notes

Several features of BACKUP vary depending on the version of DOS. With DOS versions prior to V3.3, BACKUP creates a heading file called *BACKUPID.@@@* on each diskette. This file contains the date, time, and diskette number of the backup. BACKUP stores each backed-up file individually on the diskette. Each file contains a 128-byte heading that identifies the path, file name, the file's original directory attributes, and whether the file is complete or partial.

With V3.3 and V4, BACKUP uses a different approach for the diskette that holds the backup files. A file called *CONTROL.nnn* holds the directory, file names, and other housekeeping information. All the backed-up files are placed in one large file called *BACKUP.nnn*. (For both files, the *nnn* represents the number of the backup diskette.) The two files are marked as read-only, so that you cannot inadvertently erase or alter them.

BACKUP V3.3 and V4 places all the files into a larger file and thereby gains speed over its predecessors. The backup process is reduced to less than half the time required by earlier versions. You cannot, however, determine which diskette holds a given file. The log file is the only method for locating a file with DOS V3.3 and V4.

Messages

1. `***Backing up files to target drive; d: ***`
   `diskette Number: nn`

   INFORMATION: BACKUP displays this message as the program archives the files from the source disk to the receiving floppy disk in drive d: or to the receiving hard disk drive d:. If you are using floppy disks to back up, `nn` is the backup disk's sequential number, starting with 1. After the message shown in item 1, DOS displays a list of the files that it is backing up. DOS displays the complete path and file name for each file.

2. `Cannot execute FORMAT`

   WARNING: BACKUP could not execute FORMAT because (1) FORMAT is not in the current directory or in a directory specified to the PATH command; or (2) you have insufficient memory to run FORMAT. When you see this error message, FORMAT does not run. If you have no formatted disks, abort BACKUP, format additional disks, and rerun BACKUP. If you have formatted disks, insert one into the disk drive and continue the backup process.

3. `Cannot FORMAT nonremovable drive d:`

   ERROR: The destination disk is a hard disk or networked disk drive, and you used the /F switch to request formatting.

When you see this message, BACKUP aborts. To recover from this error, issue the BACKUP command again without the /F switch or use a floppy disk drive as the destination.

4. `Disk full error writing to BACKUP Log File`
   `Strike any key to continue`

   WARNING: The disk that holds the BACKUP log file is full. Press any key to continue backing up your files. However, BACKUP will not write any new information to the log file.

5. `Error opening logfile`

   ERROR: BACKUP could not open the log file. You may have made one of the following errors: (1) you omitted the file name; (2) you gave an improperly phrased file name; (3) you gave an incorrect disk drive name (remember that you cannot place the log file on the destination disk); (4) you gave an incorrect path; (5) the log file is in use by another program; or (6) the disk's root directory has no additional directory entries available.

   If you have used the /L switch without a file name, check to see whether you have sufficient room in the root directory for the log file.

   If necessary, delete or move a file from the root directory and try again.

   If you specified a file name, make sure that the file name is correct and that you spelled and punctuated the name correctly. Then try the command again.

6. `Fixed Backup device is full`

   ERROR: The hard disk you specified to receive the backed-up files is full. To remedy the error, begin the backup process again using floppy disks, or delete unwanted files from the target hard disk. Then try BACKUP again.

7. `Insert backup diskette nn in drive d:`
   `Strike any key when ready`

   INFORMATION: This message appears when the source of the backup is a floppy disk, not a hard disk. Insert the proper disk number `nn` in drive D: and press a key to start or continue the backup process.

8. `Insert backup diskette nn in drive d:`

   ```
   Warning! Files in the target drive
   d:\ root directory will be erased
   Press any key to continue
   ```

   INFORMATION and WARNING: This message appears during the backup process when you are backing up onto floppy disks. When you use the /A switch, the message does not appear for the first target disk, but does appear for subsequent disks. This message appears for all disks when you are not using the /A switch.

The message instructs you to put the first or next backup disk (disk number `nn` in the series) into drive `d:` and press any key to continue. The message also warns that BACKUP will delete any existing files in the root directory of the receiving disk before transferring any files. Be sure that the proper disk is in the disk drive, then press any key to start.

9. `Insert last backup diskette in drive d:`
   `Strike any key when ready`

   INFORMATION: This message appears only when you are using the /A switch and invoking BACKUP without having placed the final disk for the backup set in the correct disk drive. Put the proper disk into the disk drive and press any key to start.

10. `Insert backup source diskette in drive d:`
    `Strike any key when ready`

    INFORMATION: You have specified a floppy disk drive as the source for BACKUP. BACKUP is instructing you to insert the source disk and press any key when you are ready to begin the backup process.

11. `Invalid date`

    ERROR: You made either of two errors: (1) you inserted a space between the /D switch and the date; or (2) you gave an impossible date. Make sure that the date you gave is correct and try the command again.

12. `Invalid path`

    ERROR: You made either of two errors: (1) you specified for the source drive a directory path that contains invalid characters; or (2) your directory path is too long. Check the spelling and phrasing of the PATH command and try the command again.

13. `Invalid time`

    ERROR: You made either of two errors: (1) you inserted a space between the /T switch and the time; or (2) you gave an impossible time. Check the time you gave and try the command again.

14. `Invalid drive specification`

    ERROR: You made one of the following errors: (1) you specified a nonexistent disk drive as the source or destination for BACKUP; (2) you did not specify a source disk drive name; (3) you specified a disk drive that is currently being used by an ASSIGN, SUBST, or JOIN command; or (4) you allowed a resident program such as SideKick to remain active and thus interfere with BACKUP. Check the disk drive names that you specified and inactivate any interfering memory-resident programs. If necessary, restart DOS and try the command again.

15. `Last backup diskette not inserted`

    WARNING or ERROR: This message appears when you use the /A switch and when the diskette in the disk drive is not the last diskette of a previous

series. If you used a disk previously processed by BACKUP, this message appears with the `Insert last backup disk in drive d:` message. If the disk has not been previously processed by BACKUP, BACKUP aborts.

16. `*** Last file not backed up ***`

    ERROR: The hard disk used as the recipient for the backed-up files is full. The file not backed up is the last file in the list that is displayed on-screen. BACKUP deletes this backup file. The original source file is left intact and not marked as being backed up.

    To correct the problem, you should either back up onto floppies or delete unwanted files from the target hard disk. Then try the backup again.

17. `Logging to file filename`

    INFORMATION: You have used the /L switch, and BACKUP is recording information to the log file called `filename`.

18. `No source drive specified`

    ERROR: You did not give a disk drive name for the source. Retry the command, giving a disk drive name for the source and destination.

19. `No target drive specified`

    ERROR: You did not give a disk drive name for the destination disk, or you forgot to give the source disk drive name. Retry the command and give a disk drive name for the destination.

20. `Source and target drives are the same`

    ERROR: You specified the same disk drive to be the source of the backup and to receive the backup files. Issue the command again and use a different disk drive for either the source or the target.

21. `Target can not be used for backup`

    ERROR: BACKUP could not place files on the destination disk. The disk is flawed. If you are using floppy disks, try a different one. If you are using a hard disk, try backing up to a different disk drive or restart DOS and reissue the BACKUP command.

22. `Warning! Files in the target drive`
    `d:\ directory will be erased`
    `Strike any key when ready`

    INFORMATION and WARNING: This message appears during the backup process when you are backing up onto a hard disk. You have not used the /A switch; BACKUP will erase the files in the BACKUP subdirectory. Before you proceed, check to be sure that you have selected the proper directory.

23. `Warning! No files were found to back up`

    WARNING: No files on the disk you designated for BACKUP matched the file specifications you gave. Check the spelling of the path and file names. Then get a listing of the disk's directory to learn whether the files you want to back up are there.

Batch Command

V1, V2, V3, V4—Internal

Executes one or more commands contained in a disk file.

Syntax

*dc:pathc***filename** *parameters*

dc: is the disk drive that holds the batch file.

pathc\ is the path to the batch file.

filename is the root name of the batch file.

parameters are the parameters to be used by the batch file.

Reference

See Chapter 11.

Rules for Batch Files

1. A batch file must have the extension .BAT.
2. If you do not give a disk drive name, the current disk drive is used.
3. If you do not give a path name, the current directory is used.
4. To invoke a batch file, simply type its root name. For example, to invoke the batch file OFTEN.BAT, type **OFTEN**, and press Enter.
5. DOS executes each command one line at a time. The specified parameters are substituted for the markers when the command is used.
6. DOS recognizes a maximum of ten parameters. You can use the SHIFT subcommand to get around this limitation.
7. If DOS encounters an incorrectly phrased batch subcommand when you run a batch file, DOS displays a `Syntax error` message and continues with the remaining commands in the batch file.
8. You can stop a running batch file by pressing Ctrl-Break. DOS will display this message:

   ```
   Terminate batch job (Y/N)?
   ```

   If you press Y for yes, the rest of the commands are ignored and the system prompt appears. If you press N for no, DOS skips the current command, but continues to process the other commands in the file.

9. DOS remembers which directory holds the batch file. Your batch file may cause the current directory to change at any time.
10. Because DOS remembers which diskette holds the batch file, you can change diskettes at any time. DOS will prompt you to insert the diskette that holds the batch file, if necessary. However, with DOS V3.0, if the batch file is on a diskette, you cannot remove the diskette. If you remove it, DOS displays an error message and stops processing the batch file.
11. You can make DOS execute a second batch file immediately after the first is finished. Enter the name of the second batch file as the last command in the first file. You can also execute a second batch file within the first, returning to the first file by using the CALL subcommand.
12. Batch subcommands are valid only for batch files. You cannot execute batch file subcommands as normal DOS commands.
13. You cannot redirect input or output of a batch file. However, you can use redirection in the lines within a batch file.

Rules for the AUTOEXEC.BAT File

1. The file must be named AUTOEXEC.BAT and reside in the boot disk directory.
2. The contents of the AUTOEXEC.BAT file must conform to the rules for creating batch files.
3. DOS, when booted, automatically executes the AUTOEXEC.BAT file.
4. When DOS executes the AUTOEXEC.BAT file after the computer boots, the system does not automatically request date and time. You must place DATE and TIME commands in the AUTOEXEC.BAT file.

Rules for Creating Batch Files

1. A batch file contains ASCII text. You can use the DOS command COPY, the DOS text editor, EDLIN, or another text editor to create a batch file. If you use a word-processing program, make sure that it is in programming, or nondocument, mode when you create the batch file.
2. The root name of the batch file can be from one to eight characters long and must conform to the rules for creating file names.
3. The batch file name must have the extension BAT.
4. A batch file must not have the same root name as that of a current directory program file (file ending with .COM or .EXE). Also avoid using an internal DOS command, such as COPY or DATE, as a root name. If you use a root name to name a batch file, DOS will execute the program or the command instead of the batch file.
5. You can include any valid DOS system-level command in a batch file.
6. You can enter any valid batch subcommand. (The batch subcommands are included in this Command Reference.) You also can use the parameter markers (%0-%9) environmental variables by enclosing the name in percent signs (such as %COMSPEC%).

7. To designate a file name that contains a percent symbol as part of the file name, enter the percent symbol twice. For example, to use a file called A100%. TXT, you enter **A100%%.TXT**. This rule differs from the one for parameter markers, where percent symbols precede the parameter markers, and for environmental variables, which are enclosed in percent symbols.
8. Beginning with DOS V3.3, you can prevent the display of any line from the batch file if an @ is the first character on the line.

Notes

The batch subcommands, ECHO, FOR, and PAUSE, are accepted by DOS as system commands. At the command level, however, the usefulness of PAUSE is questionable. ECHO can be useful to echo a message to the printer or file. FOR is useful to perform a repetitive task.

All DOS versions except V3.0 enable you to change the diskette that holds the batch file. With DOS V3.0, if you change the batch file diskette, DOS displays an error message and quits processing the file.

Batch Subcommand—CALL *V3.3, V4*

Runs a second batch file and returns control to the first batch file.

Syntax

CALL *dc:pathc***filename** *parameters*

dc: is the disk drive that holds the called batch file.

pathc\ is the path to the called batch file.

filename is the root name of the called batch file.

parameters are the parameters to be used by the batch file.

Reference

See Chapter 13.

Rules

1. If you do not give a disk drive name, the current disk drive is used.
2. If you do not give a path name, the current directory is used.
3. The named batch file is run as if invoked from the keyboard. Parameters are passed to the called batch file just as if the file were invoked from the keyboard.
4. The input or output of the called batch file cannot be redirected. However, the individual lines in the called batch file can use input or output redirection.
5. When the second batch file finishes, DOS executes the next line of the first batch file.

Notes

Use the CALL command to run a second batch file from another batch file. When the second is finished, DOS resumes with processing the remaining commands in the first file.

If CALL is not used to run the second batch file, DOS concludes batch file processing at the end of the second file.

You can duplicate this procedure in versions of DOS prior to V3.3 by using COMMAND in the following form:

COMMAND /C *dc:pathc***filename** *parameters*

filename is the root name of the second batch file.

Batch Subcommand—ECHO *V2, V3, V4*

Displays a message and allows or prevents the display of batch commands and other messages as DOS executes batch subcommands.

Syntax

To display a message, use the following form:

ECHO *message*

To turn off the display of commands and other batch command messages, use the form:

ECHO OFF

To turn on the display of commands and messages, use the form:

ECHO ON

To see the status of ECHO, use the form:

ECHO

message is the text of the message to be displayed on-screen.

Reference

See Chapter 13.

Rules

1. For unconditional display of a message on-screen, use the command **ECHO message**. This means that the **message** you designate will be displayed whether ECHO is on or off.
2. When ECHO is on, the batch file displays commands as DOS executes each line. The batch file also displays any messages from the batch subcommands.
3. When ECHO is off, the batch file does not display its commands as DOS executes them. The batch file also does not display messages produced by

other batch subcommands. Exceptions to this rule are the `Strike a key when ready` message generated by the PAUSE subcommand and any ECHO message command.

4. DOS starts the system with ECHO on.
5. An ECHO OFF command is active until batch processing is complete or until an ECHO ON command is encountered. If one batch file invokes another, ECHO OFF remains in effect until the final batch file is processed.
6. ECHO affects messages produced only by batch subcommands. The command does not affect messages from other DOS commands or programs.

Notes

The ECHO message is not the same as the REM message. REM is affected by an ECHO OFF command: the message on the line with the REM subcommand is not displayed if ECHO is off. The message on the line with ECHO is always displayed.

You can suppress the display of a single batch file line by using the @ as the first character in a line. When you use the line @**ECHO OFF**, the command ECHO OFF is not displayed on-screen.

To suppress the output of a command, use I/O redirection to the null device (NUL). For example, to suppress a copied file message when you are using COPY, use the following form:

COPY file1.ext file2.ext >NUL

The command output is sent to the null device and is not displayed on-screen.

Message

```
ECHO is status ON
ECHO is status OFF
```

INFORMATION: The status of ECHO displayed as either ON or OFF.

Batch Subcommand—FOR..IN..DO

V2, V3, V4

Allows iterative (repeated) processing of a DOS command.

Syntax

FOR %%*variable* **IN** (*set*) **DO** *command*

variable is a single letter.

set is one or more words or file specifications. The file specification is in the form *d:path\filename.ext*. Wild cards are allowed.

command is the DOS command to be performed for each word or file in the set.

Reference

See Chapter 13.

Rules

1. You can use more than one word or a full file specification in the set. You must separate words or file specifications by spaces or by commas.
2. %%*variable* becomes each literal word or full file specification in the set. If you use wild-card characters, FOR..IN..DO executes once for each file that matches the wild-card file specification.
3. You can use path names with FOR..IN..DO.
4. You cannot nest FOR..IN..DO subcommands (put two of these commands on the same line). You can use other batch subcommands with FOR..IN..DO.

Notes

For the FOR..IN..DO subcommand, the major change between DOS V2 and DOS V3 is the use of path names for the file specification. DOS V2 does not allow path names, but DOS V3 does.

A set also can contain literal words, separated by spaces. *set* replaces %%*variable* when the command executes.

Message

```
FOR cannot be nested
```

ERROR: More than one FOR command was found on a command line in the batch file. Edit the batch file to correct this problem and try executing the batch file again.

Batch Subcommand—GOTO *V2, V3, V4*

Jumps to the line following the label in the batch file.

Syntax

GOTO label

label is the name used for one or more characters, preceded by a colon. Only the first eight characters of the label name are significant.

Reference

See Chapter 11.

Rules

1. The label must be the first item on a line in a batch file and must start with a colon (:).
2. When the command **GOTO label** is executed, DOS jumps to the line following **label** and continues execution of the batch file.

3. A label is never executed. DOS uses the label only as the jump-to marker for the GOTO subcommand.
4. If you issue a GOTO command with a nonexistent label, DOS issues an error message and stops processing the batch file.

Message

`Label not found`

ERROR: DOS cannot find the label specified in the GOTO command. The batch file is aborted, and the system prompt reappears.

Batch Subcommand—IF *V2, V3, V4*

Allows conditional execution of a DOS command.

Syntax

IF *NOT* **condition command**

NOT tests for the opposite of the condition.

condition is what is being tested. It can be one of the following:

ERRORLEVEL number	DOS tests the exit code (0 to 255) of the program. If the exit code is greater than or equal to the number, the condition is true.
string1 = = string2	DOS compares these two alphanumeric strings.
EXIST *d:path***filename**.*ext*	DOS tests whether *d:path***filename**.*ext* is in the specified drive or path (if you give a drive name or path name) or is on the current disk drive and directory.

command is any valid DOS batch file command.

Reference

See Chapter 13.

Rules

1. For the IF subcommand, if **condition** is true, **command** is executed. If **condition** is false, **command** is skipped and the next line of the batch file is immediately executed.
2. For the IF NOT subcommand, if **condition** is false, the command is executed. If **condition** is true, **command** is skipped, and the next line of the batch file is immediately executed.

3. The only DOS programs that produce exit codes are BACKUP, FORMAT, GRAFTABL, KEYB, REPLACE, and RESTORE. Using an ERRORLEVEL condition with a program that does not leave an exit code is meaningless.
4. For **string1 = = string2**, DOS makes a literal, character-by- character comparison of the two strings. The string comparison is based on the ASCII character set, and upper- and lowercase letters are distinguished.
5. When you are using **string1 = = string2** with the parameter markers (%)-%9), neither string may be null (empty or nonexistent). If either string is null, DOS displays a syntax error message and aborts the batch file.

Message

```
Syntax Error
```

The IF subcommand was not specified correctly. Normally, this is due to a variable in the condition that contained a null value.

Batch Subcommand—PAUSE *V1, V2, V3, V4*

Suspends batch file processing until a key is pressed and optionally displays a message.

Syntax

PAUSE *message*

message is a string of up to 121 characters.

Reference

See Chapter 13.

Rules

1. The message, a series of up to 121 characters, must be on the same batch file line with the word PAUSE.
2. When DOS encounters a PAUSE subcommand in a batch file, DOS displays the optional message only if ECHO is on.
3. Regardless of the ECHO setting, DOS displays the message

   ```
   Strike a key when ready
   ```

4. DOS suspends processing of the batch file until you press any key. Afterward, DOS continues processing the batch file lines. To end processing of a batch file, press Ctrl-Break or Ctrl-C.

Batch Subcommand—REM *V1, V2, V3, V4*

Displays a message within the batch file.

Syntax

REM *message*

message is a string of up to 123 characters.

Reference

See Chapters 11 and 13.

Rules

1. REM must be the last batch file command on the line when used with the subcommands IF or FOR..IN..DO.
2. The optional message can contain up to 123 characters and must immediately follow the word REM.
3. When DOS encounters a REM subcommand in a batch file, DOS displays the message if ECHO is on.
4. The difference between the ECHO and the REM message is that the message is always displayed with ECHO. When ECHO is off, the REM message is not displayed.

Batch Subcommand—SHIFT *V2, V3, V4*

Shifts command line parameters one position to the left when a batch file is invoked.

Syntax

SHIFT

Reference

See Chapter 13.

Rules

1. When you use SHIFT, DOS moves the command line parameters one position to the left.
2. DOS discards the former first parameter (%0).

BREAK

V2, V3, V4—Internal

Determines when DOS looks for a Ctrl-Break sequence to stop a program.

Syntax

To turn on BREAK, use the following form:

BREAK ON

To turn off BREAK, use the following form:

BREAK OFF

To determine whether BREAK is on or off, use the following form:

BREAK

Reference

See Chapter 14.

Notes

The setting of BREAK depends on the type of work you will be doing on the computer. When BREAK is on, DOS checks for a Ctrl-Break sequence whenever a program asks DOS to write to a disk or screen, that is, whenever a program uses a DOS device. The Ctrl-Break key will then cause the program to halt.

With BREAK set to off, the default setting, DOS checks for a Ctrl-Break key sequence only when the program writes to the screen or the printer, reads the keyboard, or reads/writes from the serial adapter. With BREAK set to off, a user is less likely to halt a program with Ctrl-Break.

Message

`Must specify ON or OFF`

WARNING: You gave the BREAK command using a word other than ON or OFF.

CHCP

V3.3, V4—Internal

Changes or displays the code page (font) used by DOS.

Syntax

To change the current code page, use the following form:

CHCP codepage

To display the current code page, use the following form:

CHCP

codepage is a valid three-digit code page number.

Reference

See Chapter 15.

Rule

You must use NLSFUNC prior to issuing this command.

Notes

CHCP is a system-wide code page (font) changer. CHCP simultaneously resets all affected devices to the changed font. MODE works similarly but changes only one device at a time.

When you have successfully selected a code page, the new code page becomes the specified code. If you have included CONFIG.SYS directives for devices that use code pages (directives such as DEVICE=PRINTER.SYS or DEVICE=DISPLAY.SYS), CHCP loads the correct code pages for the devices.

You can access the COUNTRY.SYS file to get new country information. If you did not specify the location of COUNTRY.SYS when you invoked NLSFUNC, COUNTRY.SYS must exist in the current disk's root directory or DOS will return a `File not found` message.

Messages

1. `Code page nnn not prepared for all devices`

   ERROR: CHCP could not select the code page `nnn` because of one of the following errors: (1) you did not use MODE to prepare a code page for this device, (2) an I/O error occurred while DOS was sending the new font information to the device, (3) the device is busy (for example, a printer is in use or off-line), or (4) the device does not support code-page switching.

   Make sure that the command **MODE CODEPAGE PREPARE** has been issued for the appropriate devices and that the devices are on-line and ready. Then try CHCP again.

2. `Code page nnn not prepared for system`

   ERROR: CHCP could not select the code page `nnn` because of one of the following errors: (1) you did not run NLSFUNC, (2) you specified an invalid code page (`nnn`) for your country, or (3) you did not prepare a code page using the MODE command. Be sure that you have run NLSFUNC and that you used the command **MODE CODEPAGE PREPARE** to prepare the code page for the appropriate devices.

3. `File not found`

   ERROR: CHCP could not find the COUNTRY.SYS file. One of two errors occurred: (1) you did not specify the location of COUNTRY.SYS when you gave the NLSFUNC command, or (2) COUNTRY.SYS is not in the current disk drive's root directory.

4. `NLSFUNC not installed`

   ERROR: You attempted to change the code page but did not initiate NLSFUNC prior to using CHCP. If you plan to use code pages and have the

correct directives in CONFIG.SYS, add NLSFUNC to your AUTOEXEC.BAT file so that it will be executed when you boot the computer.

CHDIR or CD *V2, V3, V4—Internal*

Changes or shows the path of the current directory.

Syntax

To change the current directory, use either of the following forms:

CHDIR *d:***path**

CD *d:***path**

To show the current directory path on a disk drive, use either of these forms:

CHDIR *d:*

CD *d:*

d: is a valid disk drive name.

path is a valid directory path.

Reference

See Chapter 6.

Rules

1. If you do not indicate a disk drive, the current disk drive is used.
2. When you give a path name, DOS moves from the current directory to the last directory specified in the path.
3. If you want to start the move with the disk's root directory, use the back-slash (\) as the path's first character. Otherwise, DOS assumes that the path starts with the current directory.
4. If you give an invalid path, DOS displays an error message and remains in the current directory.

Examples

For the following examples, assume that a hard disk has the following directives:

```
C:\
C:\DOS
C:\WP
C:\DATA
C:\DATA\LETTERS
C:\DATA\MEMOS
```

CD, the short form of CHDIR, is used in the examples, as this is the form you are more likely to use.

1. To move to a single directory, use:

   CD \DOS

   This example moves to a single directory, starting from the root directory, moving to the directory named \DOS.

2. To move to a second-level directory, use:

   CD \DATA\MEMOS

   Starting from the root directory, DOS moves to \MEMOS.

3. To move to a directory using the shorthand double period, use:

   CD ..\LETTERS

   Starting from the \DATA\MEMOS directory, DOS moves from \MEMOS to the \LETTERS directory.

   The double period (..) stands for the parent directory of the current directory. The parent of MEMOS is DATA. Using the path ..\LETTERS has the same effect as specifying \DATA\LETTER.

4. To move from a subdirectory to the next level subdirectory, use the following form:

   CD MEMOS

   In this example, the current directory is \DATA. When you move to a single next-level directory, do not precede the directory name with a backslash.

Notes

You use CHDIR, or the short form CD, to maneuver through hierarchical directories. You can always use CD in place of CHDIR.

You have two methods for maneuvering through the hierarchical directories: (1) starting at the root (top) directory of the disk and moving down, or (2) starting with the current directory and moving in either direction.

To start at the root directory of a disk, you must begin the path with the path character (\), as in \ or **B:**\. When DOS sees the path character (\) as the first character in the path, the system starts with the root directory. Otherwise, DOS starts with the current directory.

To start the move with the current directory, use one of the subdirectory names or a parent directory symbol (..). DOS knows you are starting with the current directory. To move up one level, use the double period (..). DOS moves to the parent directory (the directory that holds the name of the current directory). A disk drive name can precede the parent directory symbol (..). If you do not use this symbol, you must use a subdirectory name that appears in the current directory.

To move through more than one directory at a time, separate each directory name with the path character (\). You can chain together as many directories as you want, provided that the total number of characters for the path does not exceed 63.

You are not restricted to changing directories on the current disk. For example, if the current drive is drive A: and the disk with the sample directory is in drive B:, you can type **B:** before each path name, and your commands will work the same way.

Message

`Invalid directory`

ERROR: A directory you specified does not exist. This error can occur for several reasons: (1) your spelling of the directory name may be incorrect, (2) you forgot or misplaced the path character (\) between the directory names, or (3) the directory does not exist in the path you specified. When this error occurs, CHDIR aborts and remains in the current directory.

CHKDSK *V1, V2, V3, V4—External*

Checks the directory and the file allocation table (FAT) of the disk and reports disk and memory status. CHKDSK can also repair errors in the directories or the FAT.

Syntax

*dc:pathc***CHKDSK** *d:path**filename.ext /F/V*

dc: is the disk drive holding the command.

pathc\ is the path to the command.

d: is the disk drive to be analyzed.

path\ is the directory path to the files to be analyzed.

filename.ext is a valid DOS file name. Wild cards are permitted.

Switches

/F Fixes the file allocation table and other problems if errors are found.

/V Shows CHKDSK's progress and displays more detailed information about the errors the program finds. (This switch is known as the *verbose* switch.)

Reference

See Chapter 5.

Rules

1. If you do not give a disk drive name, the current disk drive is used.
2. To check a diskette, make sure that the diskette you want to analyze is in the drive before you run CHKDSK. If you have a one-drive system and want to analyze a diskette that does not contain CHKDSK, specify a second drive.
3. You must direct CHKDSK to make repairs to the disk by giving the **/F** switch. CHKDSK asks you to confirm that you want the repairs made before the program proceeds.
4. If you give a file name, with or without wild cards, CHKDSK checks the file(s) for continuity.
5. Do not combine different versions of CHKDSK and DOS. Do not, for example, use CHKDSK from any increment of DOS V2 with DOS V3 or later versions or you may lose files.
6. CHKDSK will not process a directory on which you used the JOIN command—that is, a second disk joined to a subdirectory. CHKDSK processes the real portion of the disk, then displays a warning message that the command cannot process the directory on which you used JOIN. CHKDSK then continues processing the real portion of the disk/directory.
7. CHKDSK will not process a disk on which you used a SUBST command. Such a disk is a subdirectory using a disk drive name as an alias. To analyze the files in the alias subdirectory, you must give the full, real path name to the files. If C:\WORDS\LETTERS is a subdirectory on which you have used SUBST to name the subdirectory drive E:, the command **CHKDSK E:** produces an error message. If your current disk drive and directory is C:\, you must use either of the following forms:

   CHKDSK C:\WORDS\LETTERS

   CHKDSK WORDS\LETTERS
8. CHKDSK will not process a disk drive on which you used the ASSIGN command. If you gave the command **ASSIGN A=C**, you cannot analyze drive A: unless you first unassign the disk drive with ASSIGN.
9. CHKDSK will not process a networked (shared) disk. You must run CHKDSK on the computer that owns the disk drive, and you must pause (using **NET PAUSE**) network processing or disconnect (using **NET DISCONNECT**) the PC from the network before you run CHKDSK.

Examples

1. **CHKDSK**

   DOS analyzes the disk in the current drive.
2. **CHKDSK B:**

   DOS analyzes the disk in drive B:.

3. **CHKDSK /F**

   DOS analyzes the current drive and asks permission to repair the file allocation table (FAT) if CHKDSK finds a flaw. In case of a flaw, you may receive the following message:

   ```
   xxxx lost clusters found in xxx chains
   Convert lost chains to files (Y/N)?
   ```

   If you answer Y for yes, CHKDSK converts the lost disk areas into files. These files will appear in the disk's root directory, under the name `FILExxxx.CHK`, for which `xxxx` is a consecutive number between 0000 and 9999. If these files do not contain anything useful, you can delete them.

4. **CHKDSK /V**

   DOS invokes the "verbose" mode, which lists each directory and subdirectory on the disk and all files in the directories. You can redirect this output to a file or printer.

5. **CHKDSK *.***

   DOS checks all files in the current drive's current directory to see whether they are stored contiguously on the disk. One of two messages may be displayed:

   a. `All specified file(s) are contiguous`

   b. `d:path\filename.ext`
      `Contains xxx noncontiguous blocks`

   Message 5a means that you are getting good disk performance. Message 5b means that the files you specified are not stored contiguously on the disk. Message 5b is displayed for each file not stored contiguously (that is, the files are fragmented). If you are analyzing a diskette and many file names are listed, you can copy (using COPY, *not* DISKCOPY) the files to another diskette. If you are analyzing a hard disk, back up your entire hard disk, reformat it, and restore it. Another option, on a hard disk, is to use a disk optimizer to restructure files and eliminate fragmented files.

Notes

CHKDSK shows you the following items of information:

1. Volume name and creation date (only disks with volume labels).
2. Total disk space.
3. Number of files and bytes used for hidden or system files.
4. Number of files and bytes used for directories.
5. Number of files and bytes used for user (normal) files.
6. Bytes used by bad sectors (flawed disk space).
7. Bytes available (free space) on disk.

8. Bytes of total memory (RAM).
9. Bytes of free memory.

With DOS V4, CHKDSK also reports the following:

1. Total bytes in each allocation unit.
2. Total allocation units on the disk.
3. The available allocation units on the disk.

An allocation unit equates to a cluster.

CHKDSK checks a disk's directories and the FAT. The command also checks the amount of memory in the system and determines how much of that memory is free. If CHKDSK finds errors, it reports them on-screen before it makes a status report.

When you first use a disk, the individual 512-byte sections of the program or data files are stored in contiguous sectors. After you erase files or add others, DOS attempts to store a new file in any open sector. This method of file storage means that DOS may store a large file in several noncontiguous sectors on the disk. DOS slows down when it reads this file, because the system must move the disk drive's recording heads many times in order to read the entire file.

The command **CHKDSK filename** checks to see whether the specified files are stored contiguously on the disk. DOS reports any noncontiguously stored programs and how many different sections store the file(s). If CHKDSK reports many noncontiguous files on a diskette, you should format a new diskette. Use the command **COPY *.*** (not DISKCOPY) to consolidate your files from the old diskette to a new one.

CHKDSK processes only real disk drives or subdirectories, as opposed to alias or substitution disk drives. CHKDSK will not process a networked disk, a disk drive on which you have used the ASSIGN command (for example, drive A: in the command **ASSIGN A=C**), a disk drive on which you have used the JOIN command, or a disk on which you have used a SUBST command (which is actually a subdirectory of a disk).

To run CHKDSK on a networked disk, you must run the command from the computer that owns the disk drive. You must pause or disconnect the computer from the network before you run CHKDSK.

For disk drives on which you have used the ASSIGN command, you must break the assignment by using ASSIGN. You then can analyze the disk. You must break the JOIN command from a disk and its subdirectory on which you used JOIN before you can analyze the disk. You cannot use the disk drive letter given in the SUBST command for processing a subdirectory. You must give the real drive and path name to CHKDSK to analyze the subdirectory files.

For disks on which you have used JOIN or ASSIGN, CHKDSK reports the statistics of only the true disk and does not include the additional space offered by the JOIN or ASSIGN commands. For more information, see the JOIN command

in Chapter 5, and the ASSIGN and SUBST commands in Chapter 9. Also refer to the listings for these commands in this Command Reference.

CHKDSK will process a RAM (virtual) disk drive, such as RAMDRIVE.SYS, sometimes called VDISK.SYS, provided with DOS V3 and V4.

Messages

1. `All specified file(s) are contiguous`

   INFORMATION: The files you specified are stored in contiguous sectors on the disk, and you are getting the best performance from this diskette or disk subdirectory.

2. `filename contains n non-contiguous blocks`

   INFORMATION and WARNING: The file is written nonsequentially on the disk, which leads to less than optimal disk performance.

3. `filename`
   `Allocation error for file, size adjusted`

   WARNING: The `filename` has an invalid sector number in the file allocation table (FAT). The file has been truncated by CHKDSK at the end of the last valid sector.

   Check this file to verify that all information in the file is correct. If you find a problem, use your backup copy of the file. This message is usually displayed when the problem is in the FAT, not in the file.

4. `Cannot CHDIR to directory`
   `Tree past this point not processed`

   WARNING: CHKDSK was unable to change to a directory. Not all subdirectories under the current one are verified.

5. `Cannot CHDIR to root`
   `Processing cannot continue`

   WARNING: CHKDSK was unable to return to the root directory. CHKDSK cannot continue checking the remaining directories.

6. `Cannot CHKDSK a Network drive`

   ERROR: You attempted to run CHKDSK on a disk drive that is part of a network. You must run CHKDSK on the computer attached to the disk drive, and either pause or disconnect the computer from the network.

7. `Cannot CHKDSK a SUBSTed or ASSIGNed drive`

   ERROR: You attempted to run CHKDSK on a disk drive that is a subdirectory (SUBST) or a different disk drive (ASSIGN). If you are trying to analyze a subdirectory, rerun CHKDSK, giving the real disk drive and path name. You cannot analyze an assigned disk. You must break the assignment by giving the ASSIGN command without parameters and then rerun CHKDSK on the disk.

8. `directoryname`
   `Cannot recover direntry`
   `entry, processing continued`

   WARNING: CHKDSK cannot recover `direntry`, which is either the current directory entry (.) or the parent directory entry (..) in the named subdirectory, `directoryname`. This subdirectory may be so badly damaged that CHKDSK cannot recover the directory.

   If possible, copy all files from this subdirectory to another disk or subdirectory. Then erase the files in the original subdirectory and remove the subdirectory (using the RD command). Most likely, the files in the subdirectory reported to be in fault by CHKDSK are lost. You may need to restore the file from backup copies.

   If the problem is in a subdirectory of a diskette, copy all the files from the offending diskette to another. Either reformat or retire the offending diskette.

9. `CHDIR .. failed trying alternate method`

   ERROR: CHKDSK is confused and does not know how to return to a parent directory. This error is internal to CHKDSK or DOS (or both) and does not mean that your disk drive or diskette is bad.

   The best approach is to restart DOS and rerun CHKDSK. If you see this message a second time, get another copy of CHKDSK from your DOS master diskette. The copy of CHKDSK you are using is probably bad. Restart DOS and run CHKDSK a third time. If you still see this error, your disk has a serious flaw. Copy any files you can from the disk. Retire or reformat the diskette or reformat the hard disk.

10. `filename`
    `Contains invalid cluster, file truncated`

    WARNING: `filename` has a bad pointer to a section in the FAT on the disk. If you give the /F switch, the file will be truncated at the last valid sector. Otherwise, no action will be taken.

    Check this file to see whether all information is intact. If it is, usually CHKDSK can correct the problem with no loss of file information.

11. `filename`
    `Contains xxx noncontiguous blocks`

    INFORMATION: `filename` is not stored contiguously on the disk but is stored in `xxx` number of pieces. If you find that many files on a diskette are stored in noncontiguous pieces, copy the diskette to another to increase performance. If you are using a hard disk, back up the hard disk, format it, and then restore it. (Use the COPY, BACKUP, FORMAT, and RESTORE commands.)

12. `directoryname`
    `Convert directory to file (Y/N)?`

    WARNING: `directoryname` contains so much bad information that the directory is no longer usable. If you press Y, CHKDSK converts the directory into a file so that you can use DEBUG or another tool to repair the directory. If you press N, no action is taken.

    Respond with N the first time you see this message. Try to copy any files you can from this directory to another disk. Check the copied files to see whether they are usable. Then rerun CHKDSK to convert the directory into a file and try to recover the rest of the files.

13. `directoryname`
    `is joined,`
    `tree past this point not processed`

    WARNING: CHKDSK encountered a directory that is actually a disk joined to the currently processed disk. CHKDSK will not process this subdirectory, but will continue to process the remaining portion of the real disk.

14. `Disk error reading FAT x`

    WARNING: CHKDSK encountered a disk error while attempting to read information from FAT 1 or FAT 2 (shown by the number `x`). CHKDSK attempts to use the alternate FAT for the required information.

    If this message appears for a diskette, copy all the files to another diskette. Then retire or reformat the bad diskette. If this message appears for the hard disk, back up all files on the hard disk, reformat it, and then restore it.

    If the message appears for both FAT 1 and FAT 2, the diskette is unusable. Copy the files to another diskette and retire or reformat the diskette after you have removed all the information you can.

15. `Disk error writing FAT x`

    WARNING: CHKDSK encountered a disk error while attempting to put information into FAT 1 or FAT 2 (shown by the number `x`).

    If this message appears for FAT 1 or FAT 2 on a diskette, copy all the files to another diskette. Then retire or reformat the bad diskette. If this message appears for the hard disk, back up all files on the hard disk, reformat it, and then restore it.

    If the message appears for both FAT 1 and FAT 2, the diskette is unusable. Copy the files to another diskette and retire or reformat the diskette after you have removed all the information you can.

16. `Errors found, F parameter not specified`
    `Corrections will not be written to the disk`

    INFORMATION: An error was found by CHKDSK. This message tells you that CHKDSK will go through the motions to repair the disk, but will not actually make any changes to it because you did not give the /F switch.

If you see this message, you can freely answer Y to any CHKDSK message, knowing that the disk will not be changed. However, you can see what actions CHKDSK will take to correct the error.

This message also means that your disk does have problems. You must run **CHKDSK** with the **/F** switch to fix the disk.

17. `File allocation table bad drive d:`

    ERROR: There has been an error trying to access the requested file on drive `d:`. Run CHKDSK on the diskette to determine its status.

18. `filename`
    `First cluster number is invalid,`
    `entry truncated`

    WARNING: `filename`'s first entry in the FAT refers to a nonexistent portion of the disk. If you gave the /F switch, the file becomes a zero-length file (truncated).

    Try to copy this file to another diskette before CHKDSK truncates the file. You may not get a useful copy, however, and the original file will be lost.

19. `. or ..`

    `Entry has a bad attribute`
    `or Entry has a bad size or`
    `Entry has a bad link`

    WARNING: The link to the parent directory (..) or the current directory (.) has a problem. If you gave the /F switch, CHKDSK will attempt to repair the problem. This procedure is normally safe and does not carry the risk of losing files.

20. `filename`
    `has invalid cluster, file truncated`

    INFORMATION and WARNING: A part of the chain of FAT entries for `filename` points to a nonexistent part of the disk. If you gave the /F switch, the file is truncated at its last valid sector. If you did not give the /F switch, no corrective action is taken. Try to copy this file to a different disk and rerun **CHKDSK** with the **/F** switch. Part of the file may be lost.

21. `filename1`
    `Is cross linked on cluster x`
    `filename2`
    `Is cross linked on cluster x`

    WARNING: Two files—`filename1` and `filename2`—have an entry in the FAT that points to the same area (cluster) of the disk. In other words, the two files think that they own the same piece of the disk.

    CHKDSK will take no action on this problem. You must handle it yourself by completing the following steps: (1) copy both files to another diskette,

(2) delete the files from the original diskette, and (3) edit the files as necessary. Both files may contain some garbage.

22. `Insufficient room in root directory`
    `Erase files from root and repeat CHKDSK`

    ERROR: CHKDSK has recovered so many "lost" clusters from the disk that the root directory is full. CHKDSK will abort at this point.

    Examine the `FILExxxx.CHK` files. If you find nothing useful, delete them. Then rerun **CHKDSK** with the **/F** switch to continue recovering lost clusters.

23. `Insufficient memory`
    `Processing cannot continue`

    ERROR: CHKDSK needs more memory to function. The only remedy is to increase the memory size of your computer.

24. `directoryname`
    `Invalid current directory`

    WARNING: `directoryname` contains invalid information. CHKDSK will attempt to repair this directory. For more specific information about the problem with the directory, perform one of the following actions:

    a. Type **CHKDSK /V**

    b. Move into the faulty directory, if possible, and type the following:

    CHKDSK . /V

    The verbose mode will tell you more about what is wrong.

25. `Invalid sub-directory entry`

    ERROR: You specified an invalid subdirectory. It either does not exist or cannot be located. Check your specification and try again.

26. `Probable non-DOS disk`
    `Continue (Y/N)?`

    WARNING: The special byte in the FAT indicates that your disk is a DOS disk, but that the disk was not formatted, or was formatted under a different operating system, or is badly damaged.

    If you used the /F switch, press N. Recheck the disk without the /F switch and press Y in response to the prompt. See what action DOS takes. Then, if your disk is a DOS disk, run CHKDSK again with the /F switch.

    If you did not use the /F switch, press Y and watch what action DOS takes. Then decide whether you want to rerun CHKDSK /F to correct the disk.

27. `Processing cannot continue,`
    `message`

    ERROR: This error message indicates that CHKDSK is aborting because of an error; `message` tells you what the problem is. The likely culprit is

insufficient RAM to check the diskette. This message occurs most often with 64K systems. Because most systems have 256K or more, the message should be seen infrequently. If you do see this message, you may have to increase the amount of memory in your computer or check the diskette on another computer.

28. `Tree past this point not processed`

    WARNING: CHKDSK cannot continue down the indicated directory path because the system has found a bad track or because the subdirectory is actually a joined disk.

    If the fault is not a joined disk, copy all files from the disk to another floppy. The original disk may be no longer usable, and you may have lost some files.

29. `xxxxxxxxxx bytes disk space freed`

    INFORMATION: CHKDSK regained some disk space that was improperly marked as "in use." `xxxxxxxxxx` tells you how many additional bytes are now available. To free this disk space, review and delete any `FILExxxx.CHK` file that does not contain useful information.

30. `xxx lost clusters found in yyy chains`
    `Convert lost chains to files (Y/N)?`

    INFORMATION: Although CHKDSK has found `xxx` blocks of data allocated in the FAT, no file on the disk is using these blocks. They are lost clusters, which normally may be freed safely by CHKDSK if no other error or warning message is given.

    If you gave the /F switch and answer Y, CHKDSK will join each set of lost chains into a file placed in the root directory of the disk, called `FILExxxx.CHK`, in which `xxxx` is a consecutive number between 0000 and 9999. Examine the files and delete any that do not contain useful information.

    If you answer N and have used the /F switch, CHKDSK simply frees the lost chains so the disk space can be reused by other files. No files are created. If you answer Y and have not given the /F switch, CHKDSK displays the actions it can take, but does not take any action.

CLS *V2, V3, V4—Internal*

Erases the display screen.

Syntax

CLS

Reference

See Chapters 11 and 13.

Notes

This command clears all information on the screen and places the cursor at the home position in the upper left corner.

This command affects only the active video display, not memory.

If you have used the ANSI control codes to set the foreground and background, the color settings remain in effect.

If you have not set the foreground/background color, the screen reverts to light characters on a dark background.

COMMAND *V2,V3,V4—External*

Invokes a second copy of COMMAND.COM, the command processor.

Syntax

*dc:pathc***COMMAND** */E:size /P /C string*

dc: is the drive where DOS can find a copy of COMMAND.COM.

pathc\ is the DOS path to the copy of COMMAND.COM.

string is the set of characters you pass to the new copy of the command interpreter.

Switches

/E:size	Sets the *size* of the *environment*. Size is a decimal number from 160 to 32,768 bytes, rounded up to the nearest multiple of 16.
/P	Keeps this copy *permanently* in memory (until the next system reset).
/C	Passes the string of commands (the string) to the new copy of COMMAND.COM and returns to primary processor.

Reference

See Chapter 8.

Rules

1. The string in the /C option is interpreted by the additional copy of COMMAND.COM, just as though you had typed the string at the system level. The /C must be the last switch used on the line. Do not use the form **COMMAND**/*C string /P*.
2. You can exit from the second copy of the command processor by issuing the command EXIT if you have used the /P option (permanent).
3. If you issue the /P and /C switches together, the /P switch is ignored.

Notes

COMMAND is an advanced DOS command not recommended for newcomers to DOS. Consult the DOS *Technical Reference* manual for more information about this command.

COMMAND is most often used with the SHELL directive. This combination allows you to relocate COMMAND.COM from the root directory of a disk to a subdirectory.

You can use COMMAND with all versions of DOS prior to DOS V3.3 to call a second batch file from an originating batch file. For example, suppose that a batch file called BATCH1.BAT contains the following line:

```
COMMAND /C BATCH2
```

BATCH1 calls BATCH2.BAT. BATCH2.BAT executes and, after completing the last line of the batch file, returns to BATCH1 to complete the originating batch file. Using this method of calling a batch file is similar to using the CALL batch subcommand in DOS V3.3, V4.0, and V4.01.

COMP *V1, V2, V3, V4—External*

Compares two sets of disk files for differences. This command is not available with some DOS versions. If COMP is not available on your version of DOS, see the FC command.

Syntax

*dc:pathc***COMP** *d1:path1**filename1.ext1*
*d2:path2**filename2.ext2*

dc: is the disk drive holding the command.

pathc\ is the path to the command.

d1: is the drive containing the first set of files to be compared.

path1\ is the path to the first set of files.

filename1.ext1 is the file name for the first set of files. Wild cards are allowed.

d2: is the drive containing the second set of files to be compared.

path2\ is the path to the second set of files.

filename2.ext2 is the file name for the second set of files. Wild cards are allowed.

d1 and *d2* may be the same.

path1\ and *path2*\ may be the same.

filename1.ext and *filename2.ext2* may be the same also.

Special Terms:

d1:path1\filename1.ext1 is the *primary* file set.

d2:path2\filename2.ext2 is the *secondary* file set.

Reference

See Chapter 7.

Rules

1. If you do not give a drive name for a set, the current disk drive is used. This rule applies to d1: and d2:, as well as to dc:, the drive that holds the command itself.
2. If you do not give a path for a file set, the current directory of the disk drive is used.
3. If you do not enter a file name, all files for that set, whether primary or secondary, are compared, which is the same as entering *.*. However, only the files in the secondary set with names matching file names in the primary set are compared.
4. If you do not enter a drive, path, or file name, COMP prompts you for the primary and secondary file sets to compare. Otherwise, the correct diskettes must be in the correct drives if you are comparing files on diskettes. COMP does not wait for you to insert diskettes if you give both primary and secondary file names.
5. Only normal disk files are checked. Hidden or system files and directories are not checked.
6. Files with matching names but different lengths are not checked. A message is printed indicating that these files are different.
7. After 10 mismatches (unequal comparisons) between the contents of two compared files, COMP automatically ends the comparison and aborts.

Examples

1. **COMP A:IBM.LET C:IBM.LET**

   The file in the current directory on drive A:, IBM.LET, is compared to the file in the current directory on drive C:, IBM.LET.

2. **COMP *.TXT *.BAK**

   Files with the extension .TXT are compared to files that end with the extension .BAK on the current disk in the current directory.

3. **COMP A:\WORDS\LETTERS\DOUG.LET B:DOUG.LET**

   The file DOUG.LET in the subdirectory LETTERS on drive A: is compared to the file DOUG.LET in the current directory on drive B:.

4. **COMP C:\WORDS\DAVID C:\WORDS\MIKE**

   All files in the subdirectory DAVID on drive C: are compared with the files on the subdirectory MIKE, also on drive C:. Note that DAVID and MIKE must

be directories, not files. Because no file name is given, COMP assumes *.* (all files).

Notes

COMP is the utility for comparing files. Use the command to verify that files on which you used the COPY command are correct. Also, use COMP to check a known good program copy against a questionable copy. If a program once functioned properly but now acts strangely, check a good backup copy of the file against the copy you are using. If COMP finds differences, copy the good program to the disk you are using.

When you try to find the last revision of a file, look at its date and time in the directory instead of using COMP. Often, you can identify the latest file revision with this method, which is much faster than using COMP. To compare two entire diskettes on which you used DISKCOPY, use DISKCOMP instead of COMP.

Messages

1. `Compare error at offset xxxxxxxx`

   INFORMATION: The files you are comparing are not the same. The difference occurs at `xxxxxxxx` bytes from the beginning of the file. The number given is in hexadecimal format, base 16. The values for the differing bytes in the files are displayed also in hexadecimal format.

2. `EOF mark not found`

   INFORMATION or WARNING: This message is informational if you are comparing program files, but the message is a warning if you are comparing text files. The message indicates that COMP could not find the customary end-of-file marker (Ctrl-Z or 1A hex).

   This problem does not occur when you are comparing program files and some data files. The absence of an EOF is a problem for text files. COMP always compares the files on the basis of file length in the directory. Sometimes files are saved with extraneous information after the marker, with the extra bytes resulting in a compare error. Verify that the text files are intact. If the files are all right, COMP was comparing the extraneous part of the files.

3. `d:path\filename.ext - File not found`

   ERROR: The file `d:path\filename.ext` was not found. Check the spelling of the drive, path, and file names. Make sure that the correct diskette is in the disk drive and try the command again.

4. `d:path\filename.ext - File sharing conflict`

   ERROR: The file `d:path\filename.ext` is being used by another program or another computer on a network, and COMP cannot read the file. Verify that the file is not in exclusive use by another process or computer. Retry the command later.

5. `Files are different sizes`

   WARNING: You have asked COMP to compare two files of different lengths. Because COMP compares only files of the same size, COMP will skip the comparison.

6. `Invalid drive specification`

   ERROR: You gave a nonexistent disk drive name. Check the disk drives and the command line you typed and try again.

7. `d:path\filename.ext - Invalid path`

   ERROR: You specified a directory path that does not exist. Verify that your spelling is correct, the correct diskette is in the floppy disk drive, and that the specified directory path does exist. Then try the command again.

8. `10 Mismatches - ending compare`

   WARNING: COMP found ten mismatches between the two files you were comparing. COMP therefore assumes that it has no reason to continue and aborts the comparison.

Configuration Subcommand—BREAK

V3, V4—Internal

Determines when DOS looks for a Ctrl-Break or Ctrl-C to stop a program.

Syntax

BREAK = ON

BREAK = OFF

Reference

See Chapter 14.

Note

The setting for BREAK in the CONFIG.SYS file works in the same manner as setting BREAK from the DOS prompt.

Configuration Subcommand—BUFFERS

V2, V3, V4—Internal

Sets the number of disk buffers set aside by DOS in memory.

Syntax

BUFFERS = nn

If you are using DOS V4.0 or V4.01, use this form:

BUFFERS = nn,*mm* */X*

nn is the number of buffers to set, in the range of 1 to 99. If you have DOS V4, you may set a maximum of 10,000 buffers, using the /X switch.

mm is the number of sectors, from 1 to 8, that can be read or written at a time. The default is one.

Switch

/X Uses expanded memory for buffer storage.

Reference

See Chapter 14.

Notes

Buffers provide the memory for DOS to store data in transit between a disk drive and user RAM. The use of buffers is desirable, as, when reading from a disk, DOS checks first to see if the information is in a buffer. The information, if found in the buffer, is transferred to user memory—the disk is not accessed and computer time is saved. The more buffers you have, the more time you will save, because the data is more quickly accessible.

Buffers do not speed up disk access for all programs. For example, a spreadsheet program accesses the disk only when you retrieve or save a spreadsheet. A database and some word processors, on the other hand, access the disk quite often. Using more buffers for these programs will lessen the disk activity, as more data is likely to be stored in buffer memory.

Do not overdo setting buffers, however. One buffer occupies 528 bytes, so the setting of 50 buffers takes over 26K of RAM. You may find that a setting of 15 to 25 is quite adequate. Many software programs recommend buffer settings, and you should set your buffers accordingly.

If you have DOS V4.0 or V4.01, you can use the /X switch for buffer storage in expanded memory, if this type of memory is available. With expanded memory, the buffers act as a *disk cache*. Disk caches save unnecessary disk accessing, allowing as many as 10,000 buffers if you have the required 5280K of expanded memory. If, for example, you have 2M of expanded memory, a setting of 950 buffers would set aside approximately 500K for buffer space, which you should find quite adequate.

DOS V4.0 and V4.01 users also will find the *mm* setting quite useful to set the number of sectors to read ahead. Normally, files occupy more than one sector, which are often stored contiguously. Increasing the *mm* setting causes DOS to read ahead, placing additional information in buffers, so that the next time DOS reads the disk, the information already resides in a buffer, ready for transfer to user RAM—again saving computer time.

Configuration Subcommand—COUNTRY

V3, V4—Internal

Instructs DOS to modify the input and display of date, time, and field divider information.

Syntax

COUNTRY = nnn

If you are using DOS V3.3 or later, use the following form:

COUNTRY = nnn,*mmm,d:path\filenamef.extf*

nnn is the country code.

mmm is the code page.

d: is the drive that contains *filenamef.extf*

path is the path that contains *filenamef.extf*

filenamef.extf is the file that contains the country information, that is, COUNTRY.SYS.

Reference

See Chapter 15.

Rules

1. If *filenamef.extf* is not in the root directory of your hard disk, you must specify the drive and path containing *filenamef.extf*.
2. You cannot mix codepage codes with country codes. If you use the country code 001 (United States), you cannot use the code page code 863 (French Canada).

Configuration Subcommand—DEVICE

V2, V3, V4—Internal

Instructs DOS to load, link, and use a special device driver.

Syntax

DEVICE = *d:path***filename.ext** *options*

d: is the drive where DOS can find the device driver to be used.

path\ is the DOS path to the device driver.

filename.ext is the root file name and optional extension of the device driver.

options are any parameters or switches that may be used with a device driver.

Reference

See Chapter 14.

Rules

1. If a drive designation is not given, the current drive is assumed.
2. If a path is not given, the current directory is assumed.
3. If the device driver uses options and switches, be sure to designate these as needed.

Example

To install the ANSI.SYS in your CONFIG.SYS file, use this form:

```
DEVICE = C:\DOS\ANSI.SYS
```

This example assumes that ANSI.SYS is located in the drive C: \DOS subdirectory.

Notes

The device command is used to attach devices to DOS. The *device driver*, as it is called, can be a program that controls a physical device attached to the computer, such as a mouse. Or the device driver can be a program that creates a logical, or nonreal, device.

Several logical device drivers accompany and can be used with DOS. For example, ANSI.SYS is a device driver that accompanies DOS. This device driver controls screen activities, such as clearing the screen, moving the cursor, and changing screen colors. Another device driver is RAMDRIVE.SYS. This device driver uses part of your computer's RAM memory as a disk drive.

Configuration Subcommand— DRIVPARM

V3, V4—Internal

Defines or changes the parameters of a block device, such as a disk drive.

Syntax

DRIVPARM = /D:num */C /F:type /H:hds /I /N /S:sec IT:TRK*

Switches

/D:num Specifies the drive number, *num*, ranging from 0 to 255, where drive A:=0, drive B:=1, drive C:=2, etc.

/C Specifies that the drive supports *change-line*, meaning that the drive has sensor support to determine when the drive door is open. When the drive door is open, the drive is sensed as empty.

/F:type Is used to determine the type of drive. *type* is one of the following:

type	*Drive specification*
0	160K/320K/180K/360K
1	1.2M
2	720K
3	Single-density 8″ disk
4	Double-density 8″ disk
5	Hard disk
6	Tape drive
7	1.44M

Type 2 is the default if you do not specify /F.

/H:hds Is used to specify the total number of drive heads, where *hds* is a number from 1 to 99. The default for *hds* is 2.

/I Is used if you have a 3 1/2-inch drive connected internally to your floppy drive controller, but your ROM BIOS does not support a 3 1/2-inch drive.

/N Specifies that your drive or other block device is not removable (hard disk).

/S:sec Is used to specify the total number of sectors per side on the drive. *sec* can be a number from 1 to 999.

/T:trk Is used to specify the number of tracks per side of a disk or the total number of tracks per tape. *trk* can be a number from 1 to 99.

Reference

See Chapter 14.

Rule

Check all settings of the block device, type, number of heads, number of tracks, and number of sectors (if the device has change-line support) before using DRIVPARM.

Example

To specify an internal 3 1/2-inch, 1.44M drive that will be drive B: and will support change-line, use this form:

DRIVPARM /D:1 /C /F:7 /I

Note

You have great flexibility in adding block devices to your computer. Make sure that you specify the correct type, number of heads, and number of sectors. Incorrect values will cause the device to work incorrectly or not at all.

Configuration Subcommand—FCBS

V2, V3, V4—Internal

Specifies the number of DOS file control blocks that can be open simultaneously and how many are always kept open.

Syntax

FCBS = maxopen, *neverclose*

maxopen is the number of FCBs that can be open at any given time. The default is 4.

neverclose is the number of FCBs that are always open. The default value is 0.

Reference

See Chapter 14.

Rules

Refer to the documentation of the specific program requiring FCBs for correct settings.

Notes

FCBS is rarely used. DOS began using file handles rather than file control blocks, beginning with V2.0. The number of programs available that use FCBs is

diminishing. All major commercial programs available as of this printing use file handles. However, you may find a DOS V1 program using FCBs that you find helpful, and, in this case, you will need to use the FCBS setting.

Configuration Subcommand—FILES

V2, V3, V4—Internal

Specifies the number of file handles that may be open at any time.

Syntax

FILES = nnn

nnn is the number of file handles that may be open at any given time. The default value is 8.

Reference

See Chapter 14.

Rule

If you use several programs that require FILES settings, set your FILES statement to the maximum required value.

Configuration Subcommand—INSTALL

V4—Internal

Starts program from CONFIG.SYS. Valid programs to start with INSTALL are FASTOPEN, KEYB, NLSFUNC, and SHARE.

Syntax

INSTALL = *dc:pathc***filename.ext** *options*

dc: is the drive containing *filename.ext.*

pathc\ is the subdirectory where the *filename.ext* is located.

filename.ext is the name of the file, for example, FASTOPEN, KEYB, NLSFUNC, or SHARE.

options is any parameter(s) that the *filename.ext* command requires to function.

Reference

See Chapter 14.

Configuration Subcommand—LASTDRIVE

V3, V4—Internal

Sets the last valid drive letter acceptable to DOS.

Syntax

LASTDRIVE = x

x is the alphabetical character for the highest system drive. The highest default drive letter is E.

Reference

See Chapter 14.

Notes

If you use the SUBST command to assign drive letters to subdirectories, use the LASTDRIVE statement to increase the usable drive letters. Also, users of a local area network can use the LASTDRIVE statement to identify the highest letter that can be assigned as a drive letter to a subdirectory.

Configuration Subcommand—REM

V4—Internal

Places remarks or hide statements in the CONFIG.SYS file.

Syntax

REM *remark*

Reference

See Chapter 11.

Notes

You may find REM useful in the CONFIG.SYS file to document commands in the file. For example, if you place the DRIVPARM command in your CONFIG.SYS file, you may place a REM statement before the DRIVPARM command to explain what type of drive you are installing.

Another use for REM is to hide a command that you think may be causing problems with your computer. Suppose, for example, that after installing a device with the DEVICE statement in your CONFIG.SYS file, your computer begins acting strangely. Placing REM in front of the DEVICE statement deactivates the statement while the computer is turned on, yet will not erase the DEVICE statement from the CONFIG.SYS file. You may then reboot and test the computer. When the DEVICE statement has been repaired, remove the REM from the line in the CONFIG.SYS file, and reboot the computer.

Configuration Subcommand—SHELL

V3, V4—Internal

Changes the default DOS command processor.

Syntax

SHELL = *d:path***filename.ext** *parameters*

d: is the name of the drive where DOS can find the command processor to be used.

path\ is the DOS path to the command processor.

filename.ext is the root file name and optional extension of the command processor.

parameters are the optional parameters that the command processor uses.

Reference

See Chapter 14; also see COMMAND in this Command Reference.

Rules

1. If you do not specify a drive, the current drive is assumed.
2. If you do not specify a path, the current directory is used.
3. If you are using a command processor other than COMMAND.COM, make sure that the command processor has been thoroughly tested before putting it to use.

Example

To place COMMAND.COM in the \DOS directory on drive C: enter the following:

```
SHELL = C:\DOS\COMMAND.COM /P
```

Notes

Most users will not replace COMMAND.COM with another command processor. However, you may wish to reposition COMMAND.COM to a subdirectory. The SHELL command can help accomplish this task.

Configuration Subcommand—STACKS

V3.2, V3.3, V4.0, V4.01—Internal

Allots memory used to store information when a hardware interrupt occurs.

Syntax

STACKS = n,m

n is the number of stacks to allot. The default for computers using the 8088/8086 microprocessor is 0, while the default for those computers using the 80286, 80386sx and 80386 is 9.

m is the size in bytes of each stack. The default for computers using the 8088/8086 microprocessor is 0, while the default for those computers using the 80286, 80386sx and 80386 is 128.

Reference

See Chapter 14.

Notes

You will find that the default stacks are adequate for normal usage. However, if you begin receiving the error message `Fatal: Internal Stack Failure, System Halted`, then you should increase the stacks. First, increase the number of stacks. If the error persists, increase the size of each stack.

COPY

V1, V2, V3, V4—Internal

Copies files between disk drives and/or devices, either keeping the file name or changing it.

Syntax

To copy a file, use the following form:

COPY/*A/B d1:path1***filename1.ext1**/A/B*
*d0:path0***filename0.ext0**/A/B/V*

COPY /*A/B d1:path1***filename1.ext1**/A/B/V*

To join several files into one, use the following form:

COPY /*A/B d1:path1***filename1.ext1**/A/B*
*+ d0:path0***filename0.ext0**/A/B +...*

d1:, *d2:*, and *d0:* are valid disk drive names.

path1\ and *path2*\ are valid path names.

filename1.ext1 and **filename2.ext2** are valid file names. Wild cards are allowed.

The three periods (...) represent additional files in the form *dx:pathx*/**filenamex.extx**.

Special Terms:

The file being copied *from* is the *source file*. The names containing 1 and 2 are the source files.

The file being copied *to* is the *destination file*. This file is represented by a 0.

Switches

/V *Verifies* that the copy has been recorded correctly.

The following switches have different effects on the source and the destination.

For the source file:

/A Treats the file as an *ASCII* (text) file. The command copies all the information in the file up to, but not including, the end-of-file marker (Ctrl-Z). Anything after the end-of-file marker is ignored.

/B Copies the entire file (based on its size, as listed in the directory) as if it were a program file *(binary1)*. Any end-of-file markers (Ctrl-Z) are treated as normal characters, and the EOF characters are copied.

For the destination file:

/A Adds an end-of-file marker (Ctrl-Z) to the end of the *ASCII* text file after it is copied.

/B Does not add the end-of-file marker to this *binary file*.

References

See Chapters 2, 4, and 16.

Rules

For copying files with both source and destination given:

1. The source name comes first, followed by the destination name.
2. If you do not give a drive name, COPY uses the current drive.
3. If you do not give a path, COPY uses the current directory.
4. The following rules apply to the file name:
   a. You must give either a path name or a file name. Wild cards are allowed in the source file name. If you do not give a file name but give a path name for the source, DOS assumes *.*.
   b. If you do not give a destination file name, the copied file has the same name as the source file.
5. You may substitute a device name for the complete source or destination name.

6. When you copy between disk drives, COPY assumes that binary files are being copied (as though the /B switch were given).
7. When you copy to or from a device other than a disk drive, COPY assumes that ASCII files are being copied (as though the /A switch were given).
8. An /A or /B switch overrides the default settings for COPY (Rules 6 and 7).

For copying files with only one file specified:

1. The file specification you give *(d1:path1***filename1.ext1)* is the source. This specification must have one or both of the following:
   a. A valid file name. Wild cards are allowed.
   b. A drive name, a path name, or both. If only one is given, it must be different from the current drive name or path name; if both are given, at least one must be different from the current drive name or path name.
2. The source cannot be a device name.
3. The destination is the current drive and current directory.
4. The copied file(s) have the same name as the source file(s).
5. COPY assumes that binary files are being copied (as though a /B switch were given).

For concatenating files:

1. The destination file is the last file in the list unless you have a plus sign (+) before that file name. If you do not specify a destination file name, the first source name becomes the destination name.
2. If you do not give a drive name, the current drive is used.
3. If you do not give a path, the current directory is used.
4. The following rules apply to source files:
   a. You must give a valid file name. Wild cards are allowed, but their use may be dangerous. If you do not give a destination file name, the first file name is used as the destination file name.
   b. After the first file name, any additional source file specifications must be preceded by a plus sign (+).
5. The following rules apply to the destination file:
   a. You can have only one destination file specification. If you give a destination without wild cards, only one destination file is used. If you give a destination file name with wild cards, one or more destination files are used.
   b. If you do not give a destination, the first source file is used also as the destination, with the following results: (1) the first file that matches the wild-card file name is used as the destination file if you gave a wild card as part of the first source file name; and (2) the files to be joined are appended to the first source file.

Notes

The meanings of the /A and /B switches depend on their positions in the line. The /A or the /B switch affects the file immediately preceding the switch and all following the switch until another /A or /B switch is encountered. When one of these switches is used before any file name, the switch affects all following files until contradicted by another /A or /B switch.

Messages

1. `Cannot do binary reads from a device`

   ERROR: This message tells you one of two things:

   a. You used the /B switch when you tried to copy something from a binary device. When copying from a device, DOS needs a way to determine the end of the information to be transferred. This indication is made with an end-of-file marker, Ctrl-Z. Without this marker, DOS cannot know when to complete the transfer and therefore will wait forever.

   b. You used a /B (binary) switch somewhere on the command line in front of the device name, probably just after the word COPY. Use the /B switch after the file name or omit the /B completely.

2. `Content of destination lost before copy`

   WARNING: A destination file was not the first source file. The previous contents were destroyed. COPY will continue concatenating any remaining files.

3. `File cannot be copied onto itself`

   ERROR: You attempted to COPY a file back to the same disk and directory containing the same file name. This usually happens when you misspell or when you omit parts of the source or destination drive, path, or file name. Check your spelling and the source and destination names and try the command again.

4. `Invalid path or file name`

   ERROR: You gave a nonexistent directory or file name, or used the wrong directory name (a directory not in the path), or mistyped a name. COPY aborts at this point. If you used a wild card for a file name, COPY transfers all valid files before issuing the error message.

   Check to see which files are already transferred. Determine whether the directory and file names are spelled correctly and whether the path is correct. Then try again.

CTTY

V2, V3, V4—Internal

Changes the standard input and output device to an auxiliary console, or from an auxiliary console to the keyboard and video display.

Syntax

CTTY device

device is the device you want to use as the new standard input and output device. This name must be a valid DOS device name.

Reference

See Chapter 9.

Rules

1. The device should be a character-oriented device capable of both input and output.
2. Typing a colon (:) after the device name is optional.
3. Programs designed to work with the video display control codes may not function properly if you redirect them.
4. CTTY does not affect any other form of redirected I/O or piping. For example, the < (redirect from), the > (redirect to), and the | (pipe between programs) work as usual.

Notes

The CTTY command was designed so that a terminal or teleprinter, rather than the normal keyboard and video display, can be used for console input and output. This added versatility has little effect on most personal computer users.

You must specify a device that can both receive input and send output to the computer system. Using CTTY with a normal printer (an output-only device) is a mistake. DOS will wait forever for you to type commands on the printer's nonexistent keyboard. You must reset the computer before you can use it again.

Message

`Invalid device`

ERROR: You specified an invalid device name. Your spelling may be incorrect or DOS does not know the device.

DATE

V1, V2, V3, V4—Internal

Displays and/or changes the system date.

Syntax

DATE *datestring*

datestring is in one of the following forms:

mmddyy or *mmddyyyy* for the United States and Taiwan

dd-mm-yy or *dd-mm-yyyy* for Europe, the Middle East, French Canada and international English

yy-mm-dd or *yyyy-mm-dd* for East Asia and Sweden

(Consult the *MS-DOS User's Reference* to verify specific national usage.)

Do not attempt to use a "nonreal" floppy disk drive with DISKCOMP. If the floppy disk drive is part of a network or is a RAM disk (VDISK), or if the drive has been used with either the SUBST or the JOIN command, DISKCOMP issues an error message and quits. DISKCOMP ignores the effects of the ASSIGN command.

Rules

1. When you enter a correctly phrased date with this command, DOS sets the date and returns to the system prompt.
2. If you do not enter a date, DOS displays one that may or may not be correct. If the date is incorrect, enter a correctly phrased date for DOS to use as the new date. If the date is already correct, just press the Enter key and DOS will retain its displayed date.
3. You may use a hyphen (-), slash (/), or period (.) between the month and day and between the day and year.
4. Entering an incorrectly phrased or nonsense date (such as 02/29/87, 06/31/88, or 06:03:1988) causes DOS to display an error message and a request to try again.
5. Date entry and display correspond to the country setting in your CONFIG.SYS file.

Notes

When you boot DOS, it issues the DATE and TIME commands to set your system clock. If an AUTOEXEC.BAT file is on the boot disk, DOS does not display a prompt for the date or time. You may include the DATE or TIME commands in the AUTOEXEC.BAT file if you want these functions to be set when DOS is booted.

Some computers, such as the IBM Personal Computer AT and compatibles, have battery-operated clocks. After you set the clock when you first install the system, you should not need to enter the date or time until the battery wears out.

DOS uses the date and time. When you create or update a file, DOS updates the directory with the date you entered. This date shows which copy is the latest revision of the file. The DOS BACKUP command uses the date in selecting files to back up.

Message

`Invalid date`

ERROR: You gave an impossible date or used the wrong kind of character to separate the month, day, and year. This message also is displayed if you enter the date by using the keypad when the program is in cursor-control mode rather than numeric mode.

DEL *V1, V2, V3, V4—Internal*

Deletes files from the disk.

DEL is an alternative command for ERASE and performs the same functions. See ERASE for a complete description.

DIR *V1, V2, V3, V4—Internal*

Lists any or all files and subdirectories in a disk directory.

Syntax

DIR *d:path\filename.ext /P/W*

d: is the drive holding the disk you want to examine.

path is the path to the directory you want to examine.

filename.ext is a valid file name. Wild cards are permitted.

Switches

/P Pauses when the screen is full and waits for you to press any key.

/W Gives a wide (80-column) display of the file names. Information about file size, date, and time is not displayed.

Reference

See Chapter 8.

Rules

1. If you do not give a drive name, the current drive is used.
2. If you do not give a path, the current directory is used.
3. If you do not give a file name, all files in the directory are displayed.

4. The DIR command shows the contents of only one directory at a time.
5. You cannot use the DIR command on a disk drive on which you have used the ASSIGN or JOIN command. You must break the assignment before you view the directory on a drive on which you used ASSIGN. You must use the path name of the disk used in the JOIN command. You can use the DIR command on the host disk drive involved in a JOIN command.

Notes

The DIR command displays the following:

Disk volume name (if any)
Name of the directory (its complete path)
Name of each disk file or subdirectory
Number of files
Amount, in bytes, of free space on the disk

The DIR command, unless otherwise directed, shows also the following:

Number of bytes occupied by each file
Date/time of the file's creation/last update

The DIR command finds the disk files or subdirectories on the disk. This command shows only the files and subdirectories in the specified or the default directory. To see a list of all the files on a disk, use **CHKDSK /V** or **TREE /F**.

If you use the /W switch, the file and directory names are listed with five names on a line. The file date and size are not displayed, nor is the `<DIR>` for subdirectories. When you give the **DIR /W** command, you may have trouble determining which names refer to files and which refer to directories.

DIR does not report statistics for disk drives on which you have used the ASSIGN or JOIN commands. For disk drives on which you used JOIN, DIR reports the free space of the host disk drive (the disk drive to which the second disk drive is joined), but does not report the total of the two disk drives on which you have used JOIN. You must remove the ASSIGN command from the disk drive to find its amount of free space.

Message

`File not found`

ERROR: The path or file name you gave does not exist. The path may be incorrect, the file may not exist in the directory, or your spelling may be incorrect.

If you are listing a directory of another directory, you can move to the other directory, using CHDIR, and try the command again. If you still believe that the file is in the directory, enter **DIR /P** to display the complete directory and check the file name.

DISKCOMP *V1, V2, V3, V4—External*

Compares two diskettes on a track-for-track, sector-for-sector basis.

Syntax

*dc:pathc***DISKCOMP d1: d2:** */1 /8*

dc: is the disk drive holding the command.

pathc\ is the path to the command.

d1: and **d2:** are the disk drives that hold the diskettes to be compared. These drives may be the same or different.

Switches

/1 Compares only the first side of the diskette, even if the diskette or disk drive is double-sided.

/8 Compares only 8 sectors per track, even if the first diskette has a different number of sectors per track.

Reference

See Chapter 5.

Rules

1. If you are using a diskette system and do not give a drive name, the first floppy disk drive (usually drive A:) is used. If, however, you are on a hard disk and do not specify a drive name, you will receive an error message:

   ```
   Invalid drive specification
   Specified drive does not exist,
   or is non-removable
   ```

2. If you give only one valid floppy disk drive name, it is used for the comparison.
3. Giving a valid floppy disk drive name twice has the same effect as giving only one disk drive name.
4. If you give a hard disk drive name, whether valid or invalid, DOS displays an error message and does not perform the comparison.
5. When you are using one disk drive (Rules 1 through 3), DOS prompts you to change diskettes.
6. Only compatible diskettes should be compared. The two diskettes must be formatted with the same number of tracks, sectors, and sides.
7. Do not use DISKCOMP with a disk drive on which you have used the ASSIGN command. DISKCOMP ignores the effects of the ASSIGN.
8. Do not use DISKCOMP on a disk on which you have used JOIN or SUBST, or on a virtual (RAM) disk and networked disk drives. DOS displays an error message if you do.

Notes

DISKCOMP compares the contents of two compatible diskettes. The command compares the files track for track. Although you can use this command with any two compatible (or nearly compatible) diskettes, its most effective use is in comparing originals with diskettes that have been duplicated with DISKCOPY.

Using DISKCOMP to compare an original diskette with a duplicate produced with the COPY command can be meaningless. Because COPY duplicates files one at a time, the copied files may not reside in precisely the same locations on both the duplicated and the source diskettes. DISKCOMP seldom works under such conditions, because DOS reports that the diskette tracks are different, when in fact the files are identical. To compare diskettes that have been duplicated with COPY (not DISKCOPY), use the COMP command.

The switches /1, for one-side comparison, and /8, for eight-sector (DOS V1) comparison, affect only the portion that the program compares. If you compare two types of diskettes, DOS aborts the comparison or displays a number or a message indicating that the two diskettes are not the same.

Do not attempt to use a "nonreal" floppy disk drive with DISKCOMP. If the floppy disk drive is part of a network or is a RAM disk (VDISK), or if the drive has been used with either the SUBST or the JOIN command, DISKCOMP issues an error message and quits. DISKCOMP ignores the effects of the ASSIGN command.

If you use DISKCOPY with a 360K diskette on a 1.2M floppy disk drive and then compare the diskette on a 360K floppy drive, DISKCOMP may report that the diskettes are not the same. The 1.2M floppy disk drive records the information in a diskette band that is narrower than the band used by a 360K floppy disk drive. If you must run this copy on a 360K floppy disk drive, make another copy of the original diskette and use a 360K floppy disk drive. This is not true for 720K diskettes formatted on a 1.44M disk drive. 720K diskettes format correctly.

Messages

1. `Bad parameter`

   ERROR: You entered either an invalid or the wrong number of parameters. Check your command line and retry the operation. See also Messages 9 and 10.

2. `Cannot DISKCOMP to or from network drive`

   ERROR: You attempted to compare diskettes on a disk drive that is part of a network. You may use DISKCOMP only on the disk drives owned by the computer and only on the disk drives that are not part of the network. To correct this problem, use a different disk drive. Or you may want to pause or disconnect from the network and use DISKCOMP again.

3. `Compare error(s) on Track tt, side s`

   WARNING: The diskettes you are comparing are different at track number `tt, side s`. DISKCOMP does not specify which sectors are different, only that one or more sectors differ.

   If you have just used DISKCOPY on these diskettes and no problem was reported, you probably have a flaw in the second diskette. Reformat this diskette and try DISKCOPY again. Otherwise, assume that the diskettes are different.

4. `Comparing x sectors per track, s side(s)`

   INFORMATION: DISKCOMP states how many sectors per track (8, 9, or 15) and how many sides (1 or 2) the program is comparing.

5. `Compare process ended`

   INFORMATION: DISKCOMP indicates that the comparison of the diskettes is completed.

6. `disknumber diskette bad or incompatible`

   WARNING: The first or second diskette (disknumber) either is bad or is incompatible with the other diskette. The FAT of the first or second diskette indicates that the diskette is the wrong type for this comparison.

7. `Diskettes compare OK`

   INFORMATION: DISKCOMP compared the two diskettes and found that they match.

8. `Drive or diskette types not compatible`

   `Incompatible disks`

   ERROR: The disk drives or diskettes are different. The first diskette was successfully read on both sides. However, the second diskette or disk drive is not identical to the first diskette or drive.

   Remember, you cannot compare (1) HC diskettes with non-HC diskettes, (2) HC diskettes on non-HC disk drives, (3) double-sided diskettes with single-sided diskettes, (4) double-sided diskettes on single-sided disk drives, or (5) 9-sector diskettes (DOS V2 and V3) with 8-sector diskettes (DOS V1).

   Run a CHKDSK on both diskettes, using the diskette drive with the highest capacity. For each diskette, look at the number on the line that reads `bytes total disk space`. If the numbers are different for the diskettes, you cannot compare them. If the numbers are the same, the disk drive holding the second diskette is the wrong type (non-HC or single-sided) or has a hardware problem.

   If the numbers are the same, compare diskettes on the same disk drive. If this message reappears, you are either using a bad copy of DISKCOMP or

the second diskette is faulty. If this message does not appear again and the second drive is the correct type for the diskette, you have a hardware problem.

9. `Invalid drive specification`
   `Specified drive does not exist,`
   `or is non-removable`

   ERROR: One or both of the disk drive names you gave for the comparison (1) does not exist on your system, (2) is a RAM disk, (3) is a disk drive used in a JOIN or SUBST command, or (4) is a networked drive or a hard disk. You also may have omitted the colon following the disk drive letter. Check the disk drive name you entered. Make sure that the disk drive is "real" and try the command again.

10. `Invalid parameter`
    `Do not give filename(s)`
    `Command format: DISKCOMP d: d: [/1][/8]`

    ERROR: On the command line, you gave a switch that DISKCOMP does not recognize, gave a file or path name after the disk drive name, or omitted a colon after a disk drive name. Check your typing and try the command again.

11. `Drive d: not ready`
    `Make sure a diskette is inserted into`
    `the drive and the door is closed`

    WARNING: DISKCOMP is having difficulty reading the diskette in drive `d:`. The disk drive door may be open, an unformatted or non-DOS diskette may be in the drive, or the diskette may not be properly inserted in the disk drive. Check these conditions and try DISKCOMP again.

12. `Unrecoverable read error on drive x`
    `Track tt, side s`

    WARNING: Four attempts were made to read the data from the diskette in the specified drive. The error is at track number `tt, side s`. If drive `x` is the diskette holding the destination (copied) diskette, the copy is probably bad. (The diskette has a "hard" read error.) If drive `x` holds the original diskette, the diskette had a flaw when it was formatted or has developed a flaw during use.

    Run CHKDSK on the original diskette and look for the line `bytes in bad sectors`. If this line is displayed, the original diskette and the copy may actually be good. When you are formatting a diskette, FORMAT detects bad sectors and "hides" them. However, DISKCOMP does not check for bad sectors and attempts to compare the tracks even if bad sectors are present. If CHKDSK shows that the original diskette has bad sectors, the destination diskette may be good despite this message. Either way, you will want to retire the original diskette eventually.

If CHKDSK does not show any bad sectors, the original diskette has a bad spot, or the disk drive holding the diskette is faulty. The diskette is probably the source of the error. Run the diagnostics on your disk drive. If it passes this test, your diskette is faulty and should be retired.

DISKCOPY

V1, V2, V3, V4—External

Copies the entire contents of one diskette to another on a track-for-track basis, making a "carbon copy."

Syntax

*dc:pathc***DISKCOPY d1: d2:** */1*

dc: is the disk drive holding the command.

pathc\ is the path to the command.

d1: is the floppy disk drive that holds the source.

d2: is the floppy disk drive that holds the destination diskette.

Switch

/1 Copies only the first side of the diskette. This switch is not available on the Epson Equity version of DOS.

Special Terms:

The diskette you are copying from is the *source* or first diskette.

The diskette you are copying to is the *destination* or second diskette.

Reference

See Chapter 5.

Rules

1. The source and destination disk drives must be real floppy disk drives. They cannot be hard or networked disk drives, RAM disks, or disk drives on which you have used the JOIN or SUBST command. Defaulting to or giving a nonreal source or destination disk drive causes DOS to return an error message and abort the disk copy.
2. If you do not give a source disk drive name, DISKCOPY uses the default disk drive. If you give an improper source disk drive, DISKCOPY issues an error message and aborts. In other instances, DISKCOPY uses the current disk drive as the source disk drive.
3. If you give only one valid floppy disk drive name, it is used as both the source and destination disk drives for the copy, if your system has a single floppy disk drive. If your system has two floppy disk drives, giving only one valid disk drive name results in the message `Invalid drive specification error`.

4. Giving a valid floppy disk drive name twice is the same as giving only one disk drive name.
5. If you have a system with a single floppy disk drive, drive A: is used no matter what drive name you give.
6. When you copy on one floppy disk drive (Rules 2 and 5), DOS prompts you to insert the appropriate diskette and then waits for you to press any key before the DISKCOPY continues.
7. After copying, DISKCOPY displays the following prompt:

   ```
   Copy another (Y/N)?
   ```

   Press Y to copy another diskette or N to stop the program. If you respond with Y, the next copy is performed on the same disk drive as before.
8. If the destination diskette is not formatted or is formatted differently from the source diskette, DISKCOPY automatically formats the destination diskette and uses the same format as that of the source diskette. This formatting does not apply, however, to copying a double-sided diskette on a single-sided drive with the /1 switch.
9. DISKCOPY destroys any information recorded previously on the destination diskette. Do not use as the destination diskette one that contains information you want to keep.
10. To ensure that the copy is correct, run DISKCOMP on the two diskettes.
11. You cannot DISKCOPY a 360K or 1.2M diskette on a single-sided disk drive, regardless of whether you use the /1 switch. You cannot DISKCOPY a 1.2M diskette on a 360K disk drive, or a 1.44M on a 720K disk drive. The copy of a single- or double-sided (180K or 360K) diskette produced on the HC drive may not be reliable when used with a single- or double-sided disk drive.
12. DISKCOPY ignores the effects of an ASSIGN command.

Notes

DISKCOPY makes identical copies of diskettes. The command automatically evaluates the size of the source diskette and makes an exact copy of it on another disk drive. If the destination diskette is not formatted or is formatted differently, DISKCOPY reformats the diskette to the same size as the source diskette. For example, if the source diskette is double-sided with nine sectors per track, and the destination diskette is single-sided with eight sectors per track, DISKCOPY reformats the destination diskette and makes it double-sided with nine sectors per track. Then the information is copied to the destination diskette.

Always format your diskettes ahead of time. If FORMAT reports any bad sectors on a diskette, you cannot use that diskette as the destination with DISKCOPY. The command attempts to write to the destination diskette's bad sectors and does not produce a good copy.

Hard disk users should always give at least one floppy disk drive name with DISKCOPY. If the current disk is the hard disk drive and you type the following line:

C>DISKCOPY

you see this message:

```
Invalid drive specification
Specified drive does not exist,
or is non-removable
```

As described earlier in this book, diskettes produced on a high-capacity disk drive do not work with all single-sided (180K) or double-sided (360K) disk drives. This restriction includes diskettes on which you have used DISKCOPY. If you have had problems with reading 360K diskettes produced on HC disk drives, use a 360K disk drive for copying 360K diskettes with DISKCOPY. This problem does not occur with 720K diskettes produced on 1.44M disk drives, which produce reliable 720K diskettes.

Write-protect your source diskette. Write-protecting the source disk is particularly important when you use only one floppy disk drive to make a copy of a diskette. DOS periodically prompts you to change the diskette. If the source diskette is write-protected, you cannot damage it by inadvertently using it as a destination when DOS tells you to insert the destination diskette.

Write-protecting your source diskette can be important also for a system with two floppy disk drives in case you specify the wrong drive for the copy. An infrequent but fatal mistake is to put the source diskette into drive B: and the destination diskette into drive A:, and then to type the following:

DISKCOPY A: B:

Now the source and destination diskettes are reversed. If you do not stop DISKCOPY in time, the command destroys the information on the source diskette and replaces that information with whatever is on the former destination diskette.

Have available as much free RAM as possible. When you are using DISKCOPY, DOS reads into memory as much information from the source diskette as possible. Then DOS copies this information to the destination diskette and reads the next batch of information from the source diskette. The more free memory you have, the less time you will need to copy a diskette.

If you have used a diskette often, creating and deleting many files, use COPY *.* rather than DISKCOPY. DISKCOPY makes a physical image, or exact copy, of a diskette. A diskette on which many files have been created and deleted becomes *fragmented*, which means that a single file is stored here and there on the diskette. This fragmentation happens frequently with word-processing programs that automatically create backup copies of files. Fragmentation slows down file use, because DOS must read the file from many different places.

Use **CHKDSK *.*** to determine whether files on a diskette are fragmented. If CHKDSK *.* tells you that several files are noncontiguous, format a diskette and

use **COPY** *.* from the old floppy to the new one to make the files contiguous and thereby speed up file searches. If the disk contains subdirectories, use **XCOPY*.* /S/E.**

If you have subdirectories on a diskette, be sure to create the same subdirectories, using MKDIR or MD, on the new diskette and to copy their contents.

Messages

1. `Cannot DISKCOPY to or from a network drive`

   ERROR: You specified a disk drive used by a network, attached to your computer or to another one. You can use only disk drives attached to your computer for DISKCOPY, and the disk drives cannot be used by the network.

   If you "own" the disk drives and the drives are also part of a network, either pause or disconnect from the network and then rerun DISKCOPY. If you still get the message, check the drive names you gave or use a different disk drive and try the command again.

2. `Drive d: not ready`
   `Make sure a diskette is inserted into`
   `the drive and the door is closed`

   WARNING: DISKCOPY is having difficulty reading or writing the diskette in drive d:. The diskette was not properly inserted, the source diskette was not formatted or is a non-DOS diskette, or the drive door is open. Check each of these possibilities and press a key to continue.

3. `Drive types or diskette types not compatible`

   INFORMATION or ERROR: The disk drives or the diskettes are not compatible for copying. If the problem is in the disk drives, the message indicates an error. The message is informational if the problem is with the diskettes.

   For disk drives, the message indicates that the drive types you attempted to use are different and cannot handle the operation. The first diskette was successfully read, but the drive specified to make the copy (destination) diskette is not the right type. You cannot DISKCOPY HC diskettes on non-HC disk drives or copy double-sided diskettes on single-sided disk drives. See the identical message under DISKCOMP for remedial action.

   If the destination diskette was previously formatted with a capacity different from that of the source diskette, this message precedes the `Formatting while copying` message. DISKCOPY will reformat the destination diskette to match the format of the source diskette. No other action is necessary.

4. `Invalid drive specification`
   `Specified drive does not exist,`
   `or is non-removable`

   ERROR: One or both of the disk drive names you gave for the copy (1) does not exist on your system, (2) is a RAM disk, (3) is a disk drive being used in a JOIN or SUBST command, (4) is a networked drive on another computer,

or (5) is a hard disk. You also may have omitted the colon after a disk drive name. Check the disk drive names you entered. Verify that each disk drive is "real" and try the command again.

5. `Invalid parameter`
   `Do not specify filenames`
   `Command format: DISKCOPY d: d: [/1]`

   ERROR: You gave a path and/or file name with or in place of a disk drive name, or you gave a switch that DISKCOPY does not recognize. Check the command line to make sure you typed the command correctly and try the command again.

6. `Read error on drive d:`
   `Write error on drive d:`

   WARNING: DISKCOPY is having difficulty reading or writing the diskette in drive `d:`. The diskette is not properly inserted, the source diskette is not formatted or is a non-DOS diskette, or the drive door is open. Check these possibilities.

7. `SOURCE diskette bad or incompatible`
   `TARGET diskette bad or incompatible`

   WARNING: DISKCOPY has detected errors while reading the source diskette (first message) or writing the destination (target) diskette (second message). The error may be caused by bad sectors on either diskette, or the diskette may be in the wrong type of disk drive.

   Determine whether either diskette has bad sectors. If this is the problem, do not use either diskette with DISKCOPY. If the source diskette is bad, use COPY *.* to copy files from the source diskette. If the destination diskette is bad, use a different diskette or try to FORMAT the diskette again, and then use DISKCOPY. If the problem is a suspect disk drive, determine which is the good drive and use only that disk drive to DISKCOPY the disks.

8. `Target diskette is write-protected`
   `Press any key when`
   `ready...`

   WARNING: The destination (target) diskette has a write-protect tab (for 5 1/4-inch minifloppies) or is missing the write-enable tab (for 8-inch floppies).

   Before removing or adding the tab, make certain that the correct diskettes are in the disk drives and check the way you invoked the DISKCOPY command. If you used the write-protect notch to protect the wrong diskette, make sure that you DISKCOPY the intended source diskette to the right destination diskette.

9. `Unrecoverable read error on drive d:`
   `Side s, track tt`
   `Target diskette may be unusable`

   `Unrecoverable write error on drive d:`
   `Side s, track tt`
   `Target diskette is unusable`

   WARNING: These two messages indicate that the source diskette, the destination diskette, or the disk drives used for the DISKCOPY are flawed. The error occurred on drive `d:`, `side s`, `track tt`. The destination (target) diskette may not be an exact copy of the original and is therefore unusable.

   When the message indicates a read error, the source diskette and the source drive are suspect. If you have not experienced any difficulties with the disk drive, the source diskette may have a bad sector or track.

   A read error occurs on source diskettes with bad sectors. A diskette that has been formatted and has indicated bytes lost to bad sectors will also produce a read error when you DISKCOPY the diskette. DISKCOPY attempts to copy the bad sectors even though no useful information resides in them. In this case, the destination diskette is good and may be used. However, some disk space on the destination diskette is still lost to the bad sectors on the source diskette. This loss occurs because the FAT of the source diskette, which has the bad sectors marked out, is copied to the destination diskette. The corresponding sectors on the destination diskette are good but are lost for use.

   If the source diskette shows no bad sectors when CHKDSK is run, a recent flaw in the source diskette caused the bad sector. The destination diskette is probably unusable. Source diskettes with bad tracks should usually be copied with the COPY command rather than with DISKCOPY.

   The destination diskette, or the disk drive holding the destination diskette, is suspect if the message indicates a write error. If no other problems surfaced in previous disk drive use and the disk drive is in good working condition, the problem lies with the destination diskette. Exit from DISKCOPY and then FORMAT the intended destination diskette. If bad sectors show up when FORMAT analyzes the diskette, this diskette cannot be used with DISKCOPY. When FORMAT does not show any bad sectors, repeat the DISKCOPY. If further problems occur, try to DISKCOPY from drive B: to drive A: and suspect that one disk drive may be at fault.

ERASE *V1, V2, V3, V4—Internal*

Removes one or more files from the directory.

Syntax

ERASE *d:path***filename.ext**

DEL *d:path***filename.ext**

With DOS V4, you can add the /P switch:

ERASE *d:path***filename.ext /P**

DEL *d:path***filename.ext /P**

d: is the name of the disk drive holding the file(s) to be erased.

path\ is the directory of the file(s) to be erased.

filename.ext is the name of the file(s) to be erased. Wild cards are allowed.

Switch

/P With DOS V4, prompts you before erasing the file with the message `filename.ext, Delete (Y/N)?`

Reference

See Chapter 7.

Rules

1. If you do not give a disk drive name, the current disk drive is used.
2. If you do not give a path name, the current directory is used.
3. If you give a disk drive name and/or a path name but no file name, DOS assumes that the file name is *.* (all files). If you do not give a disk drive, path, and file name, DOS issues the error message:

   `Invalid number of parameters`

   or

   `Required parameter missing`

   After either of these messages, DOS aborts the program.
4. If you specify *.*, or no name for the file name (when you give a disk drive name and/or path name), DOS displays the following prompt:

   `Are you sure (Y/N)?`

   If you answer Y, all files in the specified directory are erased. If you answer N, no files are erased.
5. You cannot ERASE either the current (.) or the parent (..) directory from a subdirectory. To remove a directory, you must use RMDIR.

Notes

ERASE, or its short form DEL (delete), removes files. The directory entry for each erased file is altered so that DOS knows the file is not in use. The space occupied by each file erased from the disk is freed.

As long as you do not place more information on the diskette, you can recover the erased file with special utility programs, which are not provided with DOS. One such program is provided by the Norton Utilities. DOS's RECOVER does not recover erased files, as you might expect. RECOVER is designed only to repair a file that contains bad sectors or has a bad directory entry.

Messages

1. `Access denied`

   ERROR: You attempted to erase a file that is marked as read-only or is being used by another program or computer and thus is temporarily marked as read-only.

   If the file you intend to erase has the read-only attribute set, use the ATTRIB command to turn off the read-only flag. If this file is being used by another program or computer, wait until that other program or computer is done and then erase the file. If the file is on another computer and you do not have permission to erase the file, ask the operator to erase the file for you.

2. `File not found`

   ERROR: The file name you gave does not exist on the current or specified disk drive or directory. You also receive this message when you give an incorrect directory name as part of the file name. Check your typing of the names, make sure that the correct diskette is in the drive, and try again.

3. `Invalid drive specification`

   ERROR: You entered an ill-formed disk drive name (a colon is missing) or a disk drive name that does not exist. Check your spelling and try again.

4. `Invalid number of parameters`

   or

   `Required parameter missing`

   ERROR: You made one of the following errors: (1) you did not specify a disk drive, path, or file name, (2) you have spaces in the file specification, or (3) you have extraneous characters on the command line.

EXE2BIN

V1.1, V2, V3—External

Purpose

Changes suitably formatted .EXE files into .BIN or .COM files

Syntax

*dc:pathc***EXE2BIN** *d1:path1*/**filename1.ext1**
d2:path2/filename2.ext2

dc: is the disk drive holding the command.

pathc\ is the path to the command.

d1: is the disk drive holding the file to be converted.

path1/ is the directory of the file to be converted.

filename1 is the root name of the file to be converted.

.ext1 is the extension of the file to be converted.

d2: is the disk drive for the output file.

path2/ is the directory of the output file.

filename2 is the root name of the output file.

.ext2 is the extension of the output file.

Special Terms:

The file to be converted is the *source* file.

The output file is the *destination file.*

Rules

1. If you do not specify a drive for the source file, EXE2BIN uses the current drive as the source file.
2. If you do not specify a drive for the destination file, EXE2BIN uses the source drive.
3. When you do not specify a path, EXE2BIN uses the current disk directory.
4. You must specify a root name for the source file (the file to be converted).
5. If you do not specify a root name for the destination file, EXE2BIN uses the root name of the source file.
6. If you do not specify an extension for the source file, EXE2BIN uses the extension .EXE.
7. If you do not specify an extension for the destination file, EXE2BIN uses the extension .BIN.

8. The .EXE file must be in the correct format (following the Microsoft conventions).

Notes

EXE2BIN is a programming utility that converts .EXE (executable) program files to .COM or .BIN (binary image) files. The resulting program takes less disk space and loads faster. This conversion, however, may be a disadvantage in future versions of DOS. Unless you use a compiler-based language, you probably will never use this command. For further information about EXE2BIN, see the DOS *Technical Reference* manual.

EXIT *V2, V3, V4—Internal*

Leaves a secondary command processor and returns to the primary one.

Syntax

EXIT

Reference

See Chapter 8; also see the description for COMMAND.

Rules

This command has no effect if a secondary command processor is not loaded or was loaded with the /P switch.

FASTOPEN *V3.3, V4—External*

Keeps directory information in memory so that DOS can quickly find and use frequently needed files.

Syntax

*dc:pathc***FASTOPEN d:**=*nnn* . . .

For DOS V4, additional syntax is

*dc:pathc***FASTOPEN d:**=*(nnn,mmm)* . . . */X*

dc: is the name of the disk drive that holds the command.

pathc\ is the path to the command.

d: is the name of the disk drive whose directory information should be held in memory.

nnn is the number of directory entries to be held in memory (10 to 999).

mmm is the number of fragmented entries for the drive (1 to 999).

. . . designates additional disk drives in the form **d:**=*nnn* or **d:**=*(nnn,mmm)*.

Switch

/X Tells DOS to use expanded memory to store the information buffered by FASTOPEN.

Reference

See Chapter 14.

Rules

1. You must specify the name of the disk drive whose entries are in memory. The drive cannot be a floppy disk drive.
2. You can use FASTOPEN on an unlimited number of nonfloppy disk drives. Simply give the additional disk drives, separated by a space, on the same command line, or repeat the command for each disk drive.
3. If you do not specify *nnn*, FASTOPEN uses the value of 34.
4. If you give *nnn*, the value must be between 10 and 999, inclusively. FASTOPEN's minimum value is the maximum level of your deepest directory plus 1, or 10, whichever is greater.
5. The sum of *nnn* for all successful FASTOPEN commands cannot exceed 999.

Notes

FASTOPEN was added to DOS V3.3 and retained in DOS V4. This command caches directory information on files and can be used on any disk drive. The value of using the command on a RAM disk is questionable, however.

FASTOPEN works by keeping directory information in memory. Because disk buffers already hold the file allocation table information, FASTOPEN allows DOS to search memory for a file or a subdirectory entry, to locate quickly the corresponding FAT entry, and to open the file. If you have many files and use FASTOPEN effectively, you can increase DOS's performance significantly.

Unlike the BUFFERS command, which works when DOS reads or writes a file, FASTOPEN works when you first open a file.

As with BUFFERS, there is no predetermined best number to use. The default value of 34 works well with many installations. If your subdirectories run many levels deep or if you use many files, specifying a larger number can improve performance. However, using too large a value for *nnn* (greater than 200) slows the system. DOS spends more time examining in-memory directory entries than rereading the entries from the disk.

Messages

1. `Cannot use FASTOPEN for drive d:`

   ERROR: You have attempted to use FASTOPEN on drive d:, which is part of an ASSIGN, JOIN, or SUBST command. You cannot use FASTOPEN on these drives.

   Make sure you have given the correct disk drive name and try the command again.

2. `FASTOPEN installed`

   INFORMATION: The FASTOPEN command was successful.

3. `FASTOPEN already installed`

   ERROR: The FASTOPEN command already has been issued successfully and/or the command you gave was unsuccessful. This message may appear after another error message.

4. `Incorrect number of parameters`

   ERROR: You omitted the disk drive name given to the FASTOPEN command.

5. `Insufficient memory`

   ERROR: Insufficient memory is available for issuing the FASTOPEN command. Either remove any unneeded resident programs or modify your CONFIG.SYS file to free more RAM. Then restart DOS.

6. `Invalid drive specification`

   ERROR: You gave the name of a floppy disk drive to FASTOPEN. You can use FASTOPEN only on nonremovable disks. Check the command line and try the command again.

7. `Invalid parameter`

   ERROR: You made one of the following errors: (1) you gave a switch to FASTOPEN; (2) you forgot the equal sign between the disk drive name and the number of entries to be held; (3) you forgot to give a disk drive name; or (4) you forgot the colon after the disk drive name. Check the command line and try the command again.

8. `Same drive specified more than once`

   ERROR: You already issued the FASTOPEN command on the specified disk drive. After you give the FASTOPEN command for a disk drive, you cannot alter the number of directory entries for that drive. To alter the number, you must restart DOS and give the FASTOPEN command with the desired number of entries to hold.

9. `Too many drive entries`

   ERROR: You gave too many disk drive names to FASTOPEN. FASTOPEN was not installed for any of the disk drives named. Repeat the command and use fewer disk drive names.

10. `Too many name entries`

    ERROR: Either the sum of directory entries for all FASTOPEN commands exceeds 999 or the command for a single drive exceeds 999. FASTOPEN is not installed for the disk drives you have just named. Try the command again and reduce the number of entries or restart DOS and try the command again.

FC

V2, V3, V4—External

Compares two sets of disk files.

Syntax

*dc:pathc***FC** */A /B /C /L /LB x /N /T /W /x*
*d1:path1***filename1.ext1** *d2:path2***filename2.ext2**

dc: is the disk drive holding the command.

pathc\ is the path to the command.

d1: is the drive containing the first set of files to be compared.

path1\ is the path to the first set of files.

filename1.ext1 is the file name for the first set of files. Wild cards are allowed.

d2: is the drive containing the second set of files to be compared.

path2\ is the path to the second set of files.

filename2.ext2 is the file name for the second set of files. Wild cards are allowed.

Special Terms:

*d1:path1**filename1.ext1* is the *primary* file set.

*d2:path2**filename2.ext2* is the *secondary* file set.

Switches

/A	Abbreviates ASCII comparison displays.
/B	Forces a binary file comparison.
/C	Causes DOS to ignore the case of letters.
/L	Compares files in ASCII mode.
/LB x	Sets internal buffer to b lines.
/N	Displays line numbers for ASCII comparisons.
/T	Suppresses expansion of tabs to spaces.
/W	Compresses tabs and spaces.
/x	Sets the number of lines (1-9) to match (the default is 2).

Reference

See Chapter 7.

Rules

1. If you do not give a drive name for a set, the current disk drive is used. This rule applies to d1: and d2:, as well as to dc:, the drive holding the command itself.

2. If you do not give a path for a file set, the current directory for the drive is used.

3. If you do not enter a file name for a file set, all files for that set, primary or secondary, are compared, which has the same effect as entering *.*. However, only secondary-set files with names matching primary-set files are compared.

4. If you do not enter a drive name, path name, and file name, COMP prompts you for the primary and secondary file sets to compare. Otherwise, the correct diskettes must be in the correct drive, if you are comparing files on diskettes. COMP does not wait for you to insert diskettes if you give both primary and secondary file names.

5. Only normal disk files are checked. Hidden or system files and directories are not checked.

6. The drive names, paths, or filenames for the primary and secondary file sets may all be the same.

Notes

FC is a file comparison program similar to COMP, but more powerful (see COMP in this Command Reference). With FC, you can compare either source or binary files.

Source files are batch files, or files that contain text, such as program listings. Binary files are program files, and usually end in .COM or .EXE.

You can force FC to do a binary comparison by using the /B switch. If you do not use the /B switch, the FC command assumes that you want to compare source files and compares individual lines of the two files. In this manner, groups of lines from the two files are compared and the differences displayed.

FC performs a binary comparison by comparing every byte of the first file to the corresponding byte of the other file. Differences are printed in hexadecimal. It is possible, but not necessarily as useful, to perform a binary comparison on source files.

If you do not perform a binary comparison using /B, several other switches can be used. /C allows comparison of source files without regard to letter case. The /W switch causes DOS to ignore tabs and spaces in source files.

When FC does source file comparisons, it attempts to synchronize the files again after a mismatch. Consider, for example, the following two file fragments:

File 1:

This is a test of the emergency broadcast system.

File 2:
This is a test of the emergency broadcast system.

On a character-by-character comparison, these files would not match because of the blank line inserted in the first file fragment. FC would consider these files the same, because it would "sync up" again after the blank line, noticing that

line 3 of the first file matches line 2 of the second. By using the /X switch, you can specify how many lines to use during synchronization. The default is 3 lines, but you may specify any value from 1 to 9.

Messages

1. `fc: cannot open filename - No such file or directory`

   ERROR: One of the specified files or directories could not be located. Check your spelling, particularly of the path name, and try the operation again.

2. `fc: filename longer than filename`

   INFORMATION: The end of one comparison file was reached, but data remained in the other.

3. `fc: incompatible switches`

   ERROR: You have entered an invalid switch combination. Make sure that you have entered the switches correctly. Remember that you cannot use any other switches on the same command line with the /B switch.

4. `fc: no difference encountered`

   INFORMATION: No differences were found between the two files. They are identical.

5. `fc: out of memory`

   ERROR: FC does not have enough RAM available to complete the requested work. Remove your RAM-resident software and attempt the operation again. If the error persists, you will need more memory.

6. `Internal error`

   ERROR: An internal error has been detected in FC. Try the operation again. If the error persists, try the operation with a new copy of the FC file. If the error persists, contact your DOS vendor.

7. `Invalid argument`

   ERROR: You probably made an error in typing the command, or you entered the wrong number of parameters. Check your typing and try the operation again.

8. `Invalid number of parameters`

   ERROR: You made one of the following errors: (1) you did not specify a disk drive name, path name, or file name; (2) you have spaces in the file specification; or (3) you have extraneous characters on the command line.

9. `No space left on device`

   ERROR: You attempted to redirect output from FC to a disk file, and the disk is full. Erase some files to make room and retry the operation, or retry the operation without redirection.

10. `No such file or directory`

    ERROR: One of the files or directories you specified cannot be located. Check your spelling, particularly of the path name, and try the operation again.

11. `Resynch failed. Files are too different`

    INFORMATION: None of the file segments that FC could compare matched.

12. `Too many open files`

    ERROR: FC attempted to open the requested files but was prohibited by DOS, because not enough file handles were available. Either another program on your computer is using all available file handles, or your setting of FILES= in CONFIG.SYS is too low.

    If no other program is active on your computer, you must edit your CONFIG.SYS file and increase the number of files given to the FILES directive. Restart your system and try the command again. If another program is active, wait for it to finish and try the command again.

13. `usage: fc [/a][/b][/c][/l][/lbNN] [/w][t][/n][/NNNN] file1 file2`

    ERROR: FC cannot understand your request, so it showed you a message outlining its expectations. Chances are you mistyped the command line, or your parameters were not phrased correctly.

FIND *V2, V3, V4—External*

Displays from the designated files all the lines that match (or do not match, depending on the switches used) the specified string. This command can also display the line numbers.

Syntax

*dc:pathc***FIND** */V/C/N* "**string**" *d:path***filename.ext...**

dc: is the disk drive that holds the FIND command.

pathc\ is the path to the command.

string is the set of characters you want to find.

d: is the disk drive for the file.

path\ is the directory that holds the file.

filename.ext is the file you want to search.

Switches

/V	Displays lines not containing **string**
/C	Displays total number of lines containing **string**
/N	Displays lines containing **string**, preceded by file line number.

Reference

See Chapter 9.

Rules

1. You can use more than one file specification. All file specifications must appear after **string** and be separated by spaces.
2. If you do not give any file specifications, FIND expects information from the keyboard (standard input).
3. If you use switches with FIND, you must place them between the word **FIND** and the **string**. Most DOS commands require that you place switches at the end of the command line.
4. You must enclose **string** in double-quotes. To use the double-quote character in the string, use two double-quote characters in a row.

Notes

FIND is one of several filters provided with DOS V3 and V4. The command can find lines that contain **string** and those that do not. FIND also can number and count lines rather than display them.

This filter is useful when it is combined with DOS I/O redirection. You can redirect FIND's output to a file by using the > redirection symbol. Because FIND accepts a sequence of files to search, you do not have to redirect the input to FIND.

Messages

1. `FIND: Access denied filename`

   ERROR: The file is being used by another program or computer that has an exclusive-use lock on the file. Try the command later.

2. `FIND: File not found filename`

   WARNING: FIND cannot find the file, because (1) the file does not exist, (2) you gave a wrong drive or path, or (3) the spelling is incorrect.

   FIND interprets the a switch after **string** as a file name and gives this message; FIND continues to process other specified files.

3. `FIND: Invalid number of parameters`

   ERROR: You did not give FIND enough information. You must type at least the following:

   FIND "string"

4. `FIND: Invalid parameter x`

   WARNING: You gave FIND an incorrect switch. FIND recognizes only the /C, /N, and /V switches. FIND prints this warning message and continues.

5. `FIND: Read error in filename`

   ERROR: FIND encountered an error while reading the file named `filename`. Try FIND again. If the read error occurs a second time, copy the file to a different disk or subdirectory and try the command again. If an error occurs during the copy process, follow the directions under COPY to recover the file.

6. `FIND: Syntax error`

   ERROR: You did not phrase the command correctly. You probably failed to enclose the string in double quotation marks.

FORMAT *V1, V2, V3, V4—External*

Initializes a disk to accept DOS information and files. FORMAT also checks the disk for defective tracks and optionally places DOS on the floppy disk or hard disk.

Syntax

*dc:pathc***FORMAT d:** */S/1/8/V/B/4/N:ss/T:tt*

With DOS V4, you also may add the */V:label* and */F:size* switches:

*dc:pathc***FORMAT d:** */S/1/8/V/B/4/N:ss/T:tt /V:label /F:size*

dc: is the disk drive that holds the command.

pathc\ is the path to the command.

d: is a valid disk drive name.

Switches

/S Places a copy of the operating system on the disk so that it can be booted.

/1 Formats only the first side of the floppy disk.

/8 Formats an eight-sector floppy disk (V1).

/V Writes a volume label on the disk.

/B Formats an eight-sector floppy disk, leaving the proper places in the directory for any operating system version, but does not place the operating system on the disk.

/4 Formats a floppy disk in a 1.2M disk drive for double-density (320K/360K) use.

/N:ss Formats the disk with ss number of sectors—ss ranges from 1 to 99.

/T:ttt Formats the disk with ttt number of tracks per side—ttt ranges from 1 to 999.

/F:size Formats the disk to less than maximum capacity, with size designating one of the following values:

Drive	Allowable Values for size
160K, 180K	160, 160K, 160KB, 180, 180K, 180KB
320K, 360K	All of above, plus 320, 320K, 320KB, 360, 360K, 360KB
1.2M	All of above, plus 1200, 1200K, 1200KB, 1.2, 1.2M, 1.2MB
720K	720, 720K, 720KB
1.44M	All of above, plus 1440, 1440K, 1440KB, 1.44, 1.44M, 1.44MB

/V:label Transfers volume label to formatted disk. Replaces label with 11-character name for new disk.

Exit Codes

0 Successful completion of last format

1 Not defined

2 Not defined

3 Aborted by user (Ctrl-Break)

4 Aborted due to error

5 Aborted due to N response on a hard disk format

Reference

See Chapter 5.

Rules

1. If you do not give a disk drive name, the current disk drive is used.
2. Before you use a new diskette, you must format it. The only exceptions are when using a new diskette as the target for DISKCOPY (DISKCOPY also formats diskettes), or when you use BACKUP with the /F switch.
3. DOS checks the floppy disk drive. Unless otherwise directed through a switch, DOS formats the diskette to its maximum capacity: 2 sides if a double-sided drive is used; 9 sectors per track for normal disk drives; and 15 sectors per track for HC disk drives.
4. Some switches do not work together. For example, you cannot use the following switch combinations:

a. */V* or */S* with */B*

b. */V* with */8*

c. */N* or */T* with a 320/360K or hard disk drive

d. */1*, */4*, */8*, or */B* with the hard disk

e. */F* with */N* and */T*

5. FORMAT destroys information previously recorded on the diskette or hard disk, so do not FORMAT a diskette or hard disk that contains useful information.

6. To make a diskette usable with all versions of DOS, use the **/B** and **/1** switches. Format the diskette for any DOS version and format only one side.

7. If you give the disk a volume name, the name can be 1 to 11 characters long and contain any character that is legal in a file name.

8. If you use the /S switch to place the operating system on a disk and the current directory does not have a copy of DOS, DOS prompts you to put the DOS diskette into drive A: so formatting can proceed.

9. If DOS formats a hard disk that has a volume label, it prompts for needed information:

```
Enter current Volume Label for drive d:
```

To continue formatting disk drive d:, enter the disk drive's current volume label. If you do not enter the exact volume label, FORMAT displays the message:

```
Invalid Volume ID
Format failure
```

FORMAT then aborts.

10. If you are formatting a hard disk without a volume label, FORMAT displays the message:

```
WARNING, ALL DATA ON NON_REMOVABLE DISK
DRIVE d: WILL BE LOST!
Proceed with Format (Y/N)?
```

Press Y for yes to format the hard disk drive `d:`; N for no to abort formatting the hard disk.

11. Although the /4 switch can create a double-density disk in a 1.2M disk drive, the formatted diskette may be unreliable when used in double-density disk drives.

12. You cannot format a RAM disk (VDISK), a networked disk, or a disk drive on which you have used the ASSIGN, SUBST, or JOIN command.

Notes

You must format each diskette or hard disk before using it. FORMAT actually performs several tasks. The program sets up each track and sector on the disk so that it accepts information. Special information is recorded, such as track headers (one for each track), sector headers (one for each sector), and CRC (cyclic redundancy check) bits to ensure that the recorded information is accurate.

FORMAT, with V2, V3 and V4, always records nine sectors per track even if you have used the /B or /8 switch to specify eight sectors. DOS simply marks the ninth sector as invisible and unusable in the FAT.

As DOS completes the formatting of each track, DOS tests that track. If the track passes the test, DOS moves to the next track. DOS remembers any bad tracks and marks them as "reserved" in the FAT. DOS ignores any track marked as a bad track.

DOS then establishes the area for the disk's root directory and FAT. If DOS finds any bad tracks, the FAT is marked appropriately.

Never attempt to format a RAM disk. In some circumstances, FORMAT acts erratically when formatting a RAM disk, particularly RAMDRIVE (the DOS RAM disk program). The responses can range from a DOS display of a `Divide overflow` message to locking up of your computer. If the computer does lock up, you must turn it off, then on again. Obviously, the RAM disk's contents are lost, but no other disks or diskettes are damaged.

An undocumented switch in DOS 3.3 is /H, the hide prompt messages switch. /H suppresses the message to place a diskette in the drive and to press Enter to continue. This switch is used by BACKUP when it must format a diskette.

The /H switch is replaced by the switch /BACKUP with DOS V4. Two other undocumented switches are added: /SELECT and /AUTOTEST. /SELECT, designed for use with the SELECT program, immediately begins formatting a disk, does not show diskette statistics when formatting is complete, and does not ask to format another. The switch /AUTOTEST is similar to /SELECT, except that a volume label is not asked for, as normally happens when formatting a disk with DOS V4.

You are probably wiser not to use the undocumented switches, as they do not prompt you to insert a disk prior to formatting. You could accidentally format your hard disk, without a way to turn back. Use extreme caution with undocumented switches.

Be especially cautious when you format previously used hard disks. If the disk has a volume label, DOS asks you to enter the exact volume label. If the disk does not have a volume label, DOS displays an emphatic warning message and asks for your yes/no confirmation.

When you use a single-disk drive system (a floppy disk drive), write-protect the diskette that holds the FORMAT program. This action keeps you from erasing a good diskette inadvertently left in drive A:.

Messages

1. `Attempted write-protect violation`

   WARNING: The floppy disk you are trying to format is write-protected. If the disk is the one you want to format, remove the write-protect tab (for minifloppies) or move the write-protect tab down (for microfloppies). If the floppy disk in the drive is the wrong one, put the correct floppy disk in the drive and try the command again.

2. `Bad Partition Table`

   ERROR: FORMAT displays this message only when it formats the hard disk. FORMAT has detected that the hard disk's partition table does not have a valid DOS partition or any DOS partition at all.

   If the hard disk is new, the message indicates that you did not run FDISK on the disk. Run FDISK to establish a DOS partition and then repeat the FORMAT command. If the hard disk has been in use, loss of the partition table is a grave sign. An error has caused the disk to lose essential information at the start of the disk. Run FDISK, reestablish the DOS partition, and then rerun FORMAT. Another partition error indicates that the hard disk is in error and needs repair.

3. `Cannot format an ASSIGNed or SUBSTituted drive`

   ERROR: This disk drive is being rerouted with the ASSIGN command or is being used as part of a SUBST command. You cannot format a floppy disk in a disk drive that is part of an assignment or a substitution.

   If you are formatting a drive on which you used the ASSIGN command, clear the assignment by giving ASSIGN with no additional information. If you are formatting a drive on which you used a SUBST command, clear the substitution by entering **SUBSTd: D**, in which **d:** is the floppy disk drive. Then try formatting again.

4. `Disk unsuitable for system disk`

   WARNING: FORMAT detected on the floppy disk one or more bad sectors in the area where DOS normally resides. Because DOS must reside on a specified spot on the disk, and this portion is unusable, you cannot use the floppy disk to boot DOS.

   Try reformatting this floppy disk. Some floppy disks format successfully the second time. If FORMAT gives this message again, you cannot use this floppy disk as a boot disk.

5. `Drive letter must be specified`

   ERROR: You did not give a disk drive name to FORMAT. Check the command line and try the command again.

6. `Enter current Volume Label for drive d:`

   WARNING: You are attempting to format a hard disk that has a volume label. Enter the exact volume label to proceed with the format; if you do not want to enter a volume label, press Enter.

7. `Error reading partition table`
   `Error writing partition table`

   ERROR: FORMAT could not successfully read or write the hard disk's initial sectors. The error probably is a hardware failure. Run FDISK again. Delete and reestablish the DOS partition, Run FORMAT again. If the problem recurs, the hard disk has failed and needs to be repaired.

8. `Format failure`

   WARNING: FORMAT encountered a disk error that the program could not handle. This message usually comes after another error message and indicates that FORMAT has aborted.

   One cause of this error may be a floppy disk having bad sectors where DOS normally records the boot sector, the FAT, or the root directory. You cannot use a floppy disk or disk with flaws in any of these areas.

9. `Format not supported on drive d:`

   ERROR: The disk drive or RAM disk drive cannot be used by the DOS FORMAT command. This error should not occur if you are using IBM disk drives, but it does occur with most RAM disk programs.

   If you are using an IBM-compatible disk drive, run a different copy of FORMAT. If the message reappears, reboot your system and try again. If the message appears a third time, you may have a hardware problem.

   If your drive is not compatible, use the format program provided by your drive's vendor or simply use a different disk drive for FORMAT.

10. `Insert DOS disk in d:`
    `and strike any key when ready`

    INFORMATION: Before formatting a floppy disk with the /S switch, FORMAT reads into memory the DOS system files IO.SYS, MSDOS.SYS, and COMMAND.COM. This message indicates that DOS cannot find these files. To correct the problem, put a DOS floppy disk containing these three files into drive d: and press any key to continue.

11. `Insert new diskette for drive d:`
    `and strike ENTER when ready`

    INFORMATION: FORMAT is waiting for you to insert into drive d: the floppy disk to be formatted. After you put the correct disk in the drive, press Enter to start formatting. But before you press Enter, check the prompt again to verify that d: is the drive you are using for FORMAT and that the correct diskette is in the drive.

12. `Insufficient memory for system transfer`

    WARNING: The disk you are attempting to format has insufficient space to accept a copy of the DOS system files. Because no DOS floppy disk has a total capacity of less than 75K, this message indicates that something else is wrong. Your copy of the DOS system files may be in error, or the floppy disk may have so many bad sectors that it cannot be used.

    Remove the floppy disk from the disk drive and reinsert it; try FORMAT again. If this message reappears, try a different floppy disk (if FORMAT reports a large number of bad sectors) or restart DOS and try again. If this message appears a third time, you have either a disk drive failure or a bad floppy disk.

13. `Invalid characters in volume label`

    WARNING: At least one character in the label name you gave is invalid. Retype the name, making sure that the characters are appropriate for a volume name. The most common mistake is to use a period (.) in the volume name. DOS will ask you to try again.

14. `Invalid device parameters from device driver`

    ERROR: When DOS partitioned the hard disk, the partition started in the middle of a track rather than on a boundary. DOS cannot handle a partition starting in the middle of a track. Because FDISK does not allow DOS to start in the middle of a track, either the disk has a problem or the program you used is defective.

    Try running FDISK or the partitioning program again to make sure that the DOS partition starts on a track boundary. Then run FORMAT again.

15. `Invalid media or track 0 bad`
    `disk unusable`

    WARNING: Track 0 holds the boot record, the FAT, and the directory. This track is bad, and the floppy disk is unusable. Try formatting the floppy disk again. If the error recurs, the floppy disk cannot be used.

    This error can occur when you format 720K floppy disks as 1.44M floppy disks (if you forget to give the /N:9 switch when you formatted on a 1.44M disk drive), or when you format 360K floppy disks as 1.2M floppy disks (if you forget the /4 switch).

    This error can occur also when you format 1.2M floppy disks at lower capacities, such as 360K, and give the /4 switch. In this case, try using a floppy disk rated for double-sided, double-density use.

16. `Invalid parameter`

    ERROR: You have given an invalid disk drive name or an unrecognized switch. Check the command line and try the command again.

17. `Parameters not compatible`

    ERROR: You gave two or more switches that are not compatible. Check FORMAT's rules to see which switches you can use together. Then try running FORMAT again.

18. `Parameter not compatible with fixed disk`

    ERROR: You gave the /1, /4, /8, /B, /N, or /T switch when you attempted to format the hard disk. Be sure that you are formatting the correct disk drive. If you are formatting the hard disk, the only switches you may need are /N and /T.

19. `System transferred`

    INFORMATION: FORMAT has successfully written the DOS system files (IO.SYS and MSDOS.SYS) and COMMAND.COM onto the formatted disk.

20. `Unable to write BOOT`

    WARNING: Either the first track of the floppy disk or the DOS portion of the hard disk is bad. DOS cannot write the bootstrap loader (BOOT program) to the floppy disk or hard disk. The floppy disk or DOS portion of the hard disk cannot be used. Try reformatting. If the error occurs again, you cannot use the floppy disk or hard disk drive.

21. `Volume label (11 characters, ENTER for none)?`

    INFORMATION: FORMAT is requesting the volume name that you want to give to the newly formatted floppy disk or hard disk. Enter a valid volume name (a maximum of 11 characters) and press Enter. If you do not want a volume label, simply press Enter.

22. `WARNING! ALL DATA ON NONREMOVABLE`
    `DISK DRIVE d: WILL BE LOST`
    `Proceed with Format (Y/N)?`

    WARNING: FORMAT is warning that you are about to format a hard disk. Answer Y and press Enter if you want to format the hard disk. If you do not want to format the hard disk, press N and then press Enter.

GRAFTABL

V3, V3, V4—External

Loads into memory the tables of additional character sets to be displayed on the Color/Graphics Adapter (CGA)

Syntax

To install or change the table used by the CGA, use the following form:

*dc:pathc***GRAFTABL** *codepage*

To display the number of the current table, use the form:

*dc:pathc***GRAFTABL /STATUS**

To show the options available for use with GRAFTABL, use the form:

*dc:pathc***GRAFTABL ?**

dc: is the disk drive holding the command.

pathc\ is the path to the command.

codepage is the three-digit number of the code page for the display.

Exit Codes

0	GRAFTABL installed successfully for the first time.
1	The code page used for GRAFTABL has been successfully changed, or, if no new code page was specified, there is an existing code page.
2	GRAFTABL has been installed; no previous code page was installed or is installed.
3	Incorrect parameter, no change in GRAFTABL.
4	Incorrect version of DOS.

Reference

See Chapter 15.

Rules

1. To display legible characters in the ASCII range of 128 to 255 when you are in APA (all-points-addressable) mode on the Color/Graphics Adapter, load GRAFTABL.
2. GRAFTABL increases the size of DOS by 1,360 bytes.
3. *codepage* is the appropriate code page; codepage can be any of the following:

   437 United States
   850 Multilingual
   860 Portugal
   863 French Canada
   865 Norway and Denmark

   If no page is specified, the code 437 is assumed.
4. Unlike previous versions of GRAFTABL, this command can be used as often as necessary to change the code page displayed while a program is in APA mode.
5. GRAFTABL does not affect characters in the A/N (alphanumeric), or text, mode. GRAFTABL should be used only on systems that have the Color/Graphics Adapter.

6. After you invoke GRAFTABL, the only way to deactivate the command is to restart DOS.

Notes

The IBM Color/Graphics Adapter in graphics mode smears ASCII characters in the 128-to-255 range. National-language characters and many graphics characters are in this range. GRAFTABL produces a legible display of these ASCII characters.

GRAFTABL is useful only when your system is equipped with the Color/Graphics Adapter and when you use the Adapter in medium- or high-resolution graphics mode. Do not use the command with systems using the Monochrome Display Adapter (MDA), with the Enhanced Color/Graphics Adapter (EGA), or with the Video Graphics Array (VGA). This command, which has no effect on these displays, wastes the computer's free memory. The memory used by the program is trapped until you restart DOS.

GRAPHICS *V2, V3, V4—External*

Prints the graphics-screen contents on a suitable printer.

Syntax

*dc:pathc***GRAPHICS** *printer /R /B /LCD*

With DOS V4, you can add the name of a file containing printer information:

*dc:pathc***GRAPHICS** *printer filename /R /B /LCD /PRINTBOX:x*

dc: is the disk drive holding the command.

pathc\ is the path to the command.

printer is the type of IBM Personal Computer printer you are using. The printer can be one of the following:

COLOR1	Color printer with a black ribbon
COLOR4	Color printer with an RGB (red, green, blue, and black) ribbon, which produces four colors
COLOR8	Color printer with a CMY (cyan, magenta, yellow, and black) ribbon, which produces eight colors
COMPACT	Compact printer (not an option with V4.0 and V4.01)
GRAPHICS	Graphics printer and IBM ProPrinter
THERMALPC	Convertible Printer

With DOS V4, you also may specify the following printer:

GRAPHICSWIDE	Graphics Printer and IBM ProPrinter with 11-inch wide carriage

filename is the name of the file containing printer information (DOS V4 only). If no file name has been specified, DOS uses the name GRAPHICS.PRO.

Switches

/R	Reverses colors so that the image on the paper matches the screen—a white image on black background.
/B	Prints the background color of the screen. You can use this switch only when the printer type is COLOR4 or COLOR8.
/LCD	Prints the image as displayed on the PC Convertible's LCD display.
/PRINTBOX:x	Prints the image and uses the print box size *id* represented by *x*. This value must match the first entry of a Printbox statement in the printer profile (DOS V4 only) such as *lcd* or *std*.

Reference

See Chapter 9.

Message

```
Invalid parameter
```

ERROR: One of the following errors occurred: (1) you gave a printer type that GRAPHICS does not recognize; (2) you used the /B switch with a printer other than COLOR4 or COLOR8; (3) you gave the /LCD switch on a computer other than the PC convertible; or (4) you gave a switch that GRAPHICS does not recognize.

JOIN *V3.1, V3.2, V3.3, V4—External*

Produces a single directory structure by connecting one disk drive to a sub-directory of another disk drive.

Syntax

To connect disk drives, use the following form:

*dc:pathc***JOIN d1:** *d2:***dirname**

To disconnect disk drives, use this form of the command:

*dc:pathc***JOIN d1: /D**

To show currently connected drives, use the following form:

*dc:pathc***JOIN**

dc: is the disk drive holding the command.

pathc\ is the path to the command.

d1: is the disk drive to be connected. DOS calls this drive the *guest disk drive*.

d2: is the disk drive to which **d1:** is to be connected. DOS calls *d2:* the *host disk drive*.

dirname is a subdirectory in the root directory of *d2:*, the host drive. DOS calls **dirname** the *host subdirectory*. **dirname** holds the connection to **d1:**, the guest drive.

Switch

/D Disconnects the specified guest disk drive from its host.

Reference

See Chapter 9.

Rules

1. You must specify the guest disk drive name.
2. If you do not name a host disk drive, DOS uses the current disk drive.
3. You must specify the host subdirectory name. This subdirectory must be a level-one subdirectory; that is, it must belong to the host disk drive's root directory and not to another subdirectory.
4. With DOS V3.1 and V3.2, you cannot use the host disk drive's current directory as the host subdirectory. Later versions of DOS, however, allow the host drive's current directory to be used as the host subdirectory.
5. The host and guest disk drives must not be networked disk drives. DOS displays an error message and does not attempt to make a connection if either the host or the guest is a networked drive.
6. The host and guest disk drives must not be part of a SUBST or ASSIGN command. When you use JOIN on a disk drive that already is affected by SUBST or ASSIGN, JOIN may not always work correctly. The JOIN command does appear to work on disk drives that are affected by an ASSIGN or SUBST command. DOS gets confused, however, when it tries to access the joined disk drives. The results can include DOS commands that refuse to work, lost data that was written to the disk, or a computer that locks up. If your computer locks up, you must reboot DOS.
7. You cannot use the current disk drive as the guest disk drive. If you do, DOS responds with an error message and does not attempt to make the connection.
8. If the host subdirectory does not exist, JOIN creates one. If the subdirectory exists, it must be empty (it must have only the . and .. entries), or

DOS displays an error message and aborts the connection. When you break the connection, the newly created subdirectory remains. You can use this subdirectory later for another JOIN command, or delete the subdirectory by issuing the RMDIR command.

9. When the disk drives are joined by a JOIN command, the guest drive's root directory and entire directory tree are added to the host subdirectory. All subdirectories of the guest's root directory become subdirectories of the host subdirectory.

10. While a guest disk drive is joined, it appears to be part of the host subdirectory. You can access this disk drive only through the host disk drive and subdirectory. If you attempt to access the guest disk drive by using the guest's normal drive name, DOS responds with an `Invalid drive specification` error message and ignores your request.

11. To break the connection, specify the guest drive's normal name with the /D switch. You can use the guest drive's normal name only when you disconnect the drives.

12. If you want to see all the current disk drive connections, type **JOIN** with no parameters. If no connections exist, JOIN does not display any message and the system prompt appears.

13. Do not use BACKUP, RESTORE, FORMAT, DISKCOPY, or DISKCOMP on the guest or host disk drive. The command may ignore the JOIN or may issue an error message. BACKUP backs up the joined disk drives as though the two drives were one, which is not a problem until you use RESTORE. If the two disk drives are not joined at the same directory, RESTORE puts the former guest disk drive's files on the host disk drive. If the host disk drive does not have room for all the files, RESTORE issues a `Disk full` message and does not restore the remaining files. You must delete the "guest" files and rerun RESTORE as many times as necessary to bring back the additional files. This additional work is burdensome and frustrating. You can avoid this problem by not using BACKUP on a disk that is joined to another disk drive.

14. While JOIN is in effect, the DIR command works normally but reports the `bytes free` only for the host disk drive.

15. While JOIN is in effect, CHKDSK will process the host disk drive but will not process or report information on the guest portion of the drive. If you want to run CHKDSK on the guest disk drive, you must first disconnect it from the host drive.

Examples

For these examples, assume that drive C: has the following directory structure and that all subdirectories have files except \MNT, which is an empty subdirectory that holds only the **.** and **..** entries.

\WORDS
\BIN
\LANG
\MNT

1. **JOIN A: C:\MNT**

   JOIN connects the disk in drive A: to the subdirectory \MNT on drive C:.

2. **JOIN B: C:\DRIVEB**

   JOIN connects the disk in drive B: with the new subdirectory \DRIVEB on drive C:. Because \DRIVEB did not exist before the JOIN command was issued, JOIN creates this subdirectory.

3. **JOIN**

   JOIN displays the current assignments. If the preceding examples were in effect, DOS would display this message:

```
A: => C:\MNT
B: => C:\DRIVEB
```

4. **JOIN B: /D**

   JOIN disconnects the disk in drive B: from drive C:. The subdirectory \DRIVEB, which was created by JOIN, remains.

Notes

The JOIN command, which was introduced in DOS V3.1, combines two disks into a single disk. The guest disk logically (apparently) becomes a subdirectory to the host disk.

You can use JOIN to connect a RAM disk to a "real" disk, enabling the RAM disk to be used as though it were part of a floppy disk or hard disk drive. You also can use JOIN to connect two hard disk drives.

Some programs allow only one disk drive to hold data or certain parts of the program itself. Programs that were written for DOS V2 and later, however, allow subdirectory names to be specified. If you are using such a program, you can invoke the JOIN command to trick the program into using one "large" disk drive.

JOIN does not affect the guest disk drive. Rather, it affects only the way you access the files on that disk drive. You cannot exceed the maximum number of files in the guest disk's root directory. In the host subdirectory, a single file's size cannot exceed the guest disk's size. Similarly, a single file cannot span more than one disk.

You should use the JOIN command with caution. When a JOIN is active, certain DOS commands work differently and some DOS commands should not be used at all. While joined, the guest disk drive appears to be a host subdirectory. You cannot issue a DIR or CHKDSK command using the guest disk drive's let-

ter. You first must break the connection, then use DIR or CHKDSK on the former guest.

Messages

1. `Cannot JOIN a network drive`

   ERROR: You tried to connect a networked disk drive to another disk drive. If you are not trying to use a networked drive, check your spelling and try the command again.

2. `Directory not empty`

   ERROR: You tried to use a host subdirectory that is not empty; that is, it contains files other than the . and .. entries. You can either: (1) delete all files in the host subdirectory; (2) specify an empty subdirectory; (3) create a new subdirectory; or (4) name a nonexistent host subdirectory. Then try the command again.

3. `Incorrect number of parameters`

   ERROR: You omitted the guest disk drive name or the host disk drive name and subdirectory name. Check the command line and try again.

4. `Invalid parameter`

   ERROR: One of the following errors occurred: (1) you omitted the colon after a disk drive name; (2) you named a nonexistent disk drive; (3) you failed to name a host subdirectory or omitted the path character (\) before the name; (4) you named a subdirectory that does not exist in the root directory; or (5) you named a subdirectory or host disk drive when you used the /D disconnect switch. Otherwise, you specified the host disk drive's current directory as the intended host subdirectory.

KEYB *V2, V3, V4—External*

Changes the keyboard layout and characters to one of five languages.

Syntax

To change the current keyboard layout, use the following form:

*dc:pathc***KEYB** *keycode, codepage, d:path***KEYBOARD.SYS**

To specify a particular keyboard identification code, use this form:

*dc:pathc***KEYB** *keycode, codepage, d:path***KEYBOARD.SYS** **/ID:***code*

To display the current values for KEYB, use the following form:

*dc:pathc***KEYB**

dc: is the disk drive holding the command.

pathc\ is the path to the command.

keycode is the two-character keyboard code for your location.

codepage is the three-digit code page that will be used (437, 850, 860, 863, or 865).

*d:path***KEYBOARD.SYS** is the drive and path to the KEYBOARD.SYS file.

/ID:*code* is the code for the type of Enhanced Keyboard that you want.

Exit Codes

0	KEYB ran successfully.
1	Invalid keycode type, code page, or other syntax error.
2	Bad or missing KEYBOARD.SYS file.
3	KEYB could not create a keyboard table in memory.
4	KEYB could not communicate successfully with CON (the console).
5	The specified code page has not been prepared.
6	The internal translation table for the selected code page could not be found; the *keycode* and *codepage* are incompatible.
7	Incorrect version of DOS (DOS V4)

Reference

See Chapter 15.

Rules

1. To use one of the foreign-language character sets, load the KEYB program and use the appropriate two-letter code for your country.
2. Do not mix the KEYB programs from different versions of DOS. The computer will act erratically, and you will have to reboot DOS.
3. If you do not specify a *codepage*, DOS uses the default code page for your country. The default code page is established by the COUNTRY directive in CONFIG.SYS or, if the COUNTRY directive is not used, the DOS default code page. KEYB attempts to activate the default code page.
4. You must specify a code page that is compatible with your keyboard code selection. If you use a combination that is not allowed, the system acts erratically until you either use a correct KEYB combination or restart DOS.
5. If you do not specify the keyboard definition file (KEYBOARD.SYS), DOS looks for this file in the current disk's root directory. Otherwise, DOS uses the full file name to search for the file. If you do not specify a disk drive, DOS searches the current disk drive. If you do not specify a path, DOS searches the current directory. The file name (KEYBOARD.SYS) must be included.

6. After the program is loaded, it reconfigures the keyboard into the appropriate layout for the specified language. You can find information on individual layouts in your computer's operating guide, or in the reference manual that comes with DOS V3.3 or later versions.
7. To use the American English layout after you have issued the KEYB command, press Ctrl-Alt-F1 (press and hold down the Ctrl and Alt keys while you press the F1 key). To return to the foreign-language layout, press Ctrl-Alt-F2.
8. When it is used the first time, the KEYB command increases the size of DOS by approximately 2K. After that, you can use KEYB as often as you want without further enlarging DOS.
9. To display the active keyboard and the display code pages, type **KEYB** without any other parameters.

Notes

Previously, KEYB was a set of five programs, featuring one program for each of the five national languages that DOS supported. DOS V3.3 codified the five programs into one program.

KEYB hooks into the computer's ROM BIOS. Once loaded, the program alters the keyboard layout, enabling you to type in foreign-language characters. KEYB also will change the video display when the DEVICE=DISPLAY.SYS command has been included in CONFIG.SYS and if you have issued the appropriate MODE commands to prepare and select a code page for your display.

When KEYB is active, it reassigns some alphanumeric characters to different keys and introduces new characters. The new layout and characters vary among the different supported languages.

Some of the new characters are symbols or punctuation marks, such as the United Kingdom's pound sterling sign or the Spanish inverted question mark. Most new characters have *diacritics*, such as the acute accent, grave accent, circumflex, or umlaut. You can produce some characters with one keystroke. You produce an accent mark by pressing and holding down the acute or grave accent key and typing the appropriate vowel. To get the accent, press and release the acute or grave accent key, then tap the space bar.

To accommodate certain programming characters, such as the path character (\), some keys produce three different characters. The characters are created by shifting or not shifting the key or by holding down the Ctrl and Alt keys while pressing the character key. If you use the French Canadian keyboard, hold down the Alt and Shift keys and press the desired character.

If you want to see the keyboard layouts for foreign languages, consult your computer's operating manual. For older machines that do not have this information, ask your computer dealer or supplier for a copy of the appropriate pages.

Messages

1. `Active code page not available from CON device`

   INFORMATION: You issued the KEYB command to display the current setting, but the command could not determine what code page is being used. Either the DEVICE=DISPLAY.SYS directive has not been given in CONFIG.SYS, or there is no currently loaded CON code page.

   If the DISPLAY.SYS line has been included in your CONFIG.SYS file, you must give the MODE CON CODEPAGE PREPARE command to load the font files into memory. See the MODE command for more information.

2. `Bad or missing Keyboard Definition File`

   ERROR: Either the keyboard definition file, which is usually KEYBOARD.SYS, has been corrupted or KEYB could not find the file. If you did not specify a disk drive and path name, KEYB looks for the file in the current disk drive's root directory.

   Either copy the file to the root directory or give the full disk drive and path name for the file to KEYB. If this message reappears, the copy of KEYBOARD.SYS is bad. Copy the KEYBOARD.SYS file from one of your DOS disks to the root directory of the boot disk and try the command again.

3. `Code page requested (codepage) is not valid for given keyboard code`

   ERROR: You gave a keyboard code to KEYB but you did not give a code page. The specified keyboard code does not match the currently active code page (`codepage`) for the console. KEYB does not alter the current keyboard or code page. Either choose a new console code page that matches the keyboard code (by using the MODE CON CODEPAGE SELECT command), or specify the appropriate matching code page when you reissue the KEYB command.

4. `Code page specified is inconsistent with the selected code page`

   WARNING: You have specified a keyboard code and a code page to KEYB, but a different code page was active for the console (CON). The code page specified to KEYB is now active for the keyboard but not for the video display. You may notice that the characters you type do not make sense on-screen, because the keyboard and screen do not always "agree" on the characters you type.

   Use the MODE CON CODEPAGE SELECT command to activate the correct code page (the one specified to KEYB) for the video screen.

5. `Code page specified has not been prepared`

   ERROR: The DEVICE=DISPLAY.SYS directive has been included in your CONFIG.SYS file, but your KEYB command specified a keyboard code that

needs a code page that has not been prepared. Use the MODE CON CODEPAGE PREPARE command to prepare the appropriate code page for the keyboard code you want to use.

6. `Current CON code page: codepage`

   INFORMATION: The console's current code page is designated by the number `codepage`.

7. `Current keyboard code: keycode code page: codepage`

   INFORMATION: The current keyboard code is the two-character `keycode`, and the code page used by the keyboard is the three-digit `codepage`.

8. `Invalid code page specified`

   ERROR: You specified a nonexistent code page. Check the number you used for the code page and try the command again.

9. `Invalid keyboard code specified`

   ERROR: You specified a nonexistent keyboard code. Check the two-character keyboard code and reissue the command.

10. `Invalid syntax`

    ERROR: KEYB does not recognize your phrasing of the command. You probably committed one of the following errors: (1) you omitted part of the command (for example, you gave the code page but forgot the keyboard code); (2) you forgot part of the punctuation for the KEYBOARD.SYS file; or (3) you forgot to place a space or comma between the keyboard code, the code page, and the full file name of the KEYBOARD.SYS file.

11. `KEYB has not been installed`

    INFORMATION: You used the KEYB command without any parameters (this form of the command requests the current status of KEYB), and KEYB reported that it has not yet been installed. If KEYB has been installed, give the command again with the appropriate parameters.

12. `One or more CON code pages invalid for given keyboard code`

    WARNING: You used the MODE command to prepare several code pages for the console (CON), but you gave a keyboard code that is not compatible with one or more console code pages. KEYB creates the necessary information to work with those keyboard and code pages that are compatible. DOS ignores the incompatible keyboard and code page combinations.

13. `Unable to create KEYB table in resident memory`

    ERROR: When KEYB is first installed, it fits within a predetermined amount of RAM. You used other commands to add extra code pages to the console, and you tried to use KEYB again. KEYB, which cannot expand past the pre-

determined amount of RAM, could not adapt to the additional code pages. If you must use the additional code pages, restart DOS and give any needed MODE code page commands before you issue KEYB.

LABEL

V3, V4—External

Creates, changes, or deletes a volume label for a disk. This command is not available with the Epson Equity implementation of DOS.

Syntax

*dc:pathc***LABEL** *d:volume_label*

dc: is the disk drive holding the command.

pathc\ is the path to the command.

d: is the disk drive whose label will be changed.

volume_label is the disk's new volume label.

Reference

See Chapter 5.

Rules

1. If you do not name a disk drive, DOS uses the current disk drive.
2. If you give a valid volume label, it immediately becomes the volume label for the specified disk drive.
3. If you do not specify a volume label, DOS prompts you to enter a new volume label. You may do one of the following:
   a. Enter a valid volume name and press Enter. DOS makes this name the new volume label. If a volume label already exists, DOS replaces it with the new volume label.
   b. Press Enter to delete the current label without specifying a replacement label. DOS asks you to confirm the deletion.
4. If you enter an invalid volume label, DOS responds with a warning message and asks again for the new volume label.
5. Do not use LABEL on a networked disk drive (one that belongs to another computer). If you try to label a networked drive, DOS displays an error message and ignores the command.
6. Do not use LABEL on a disk in any drive that is affected by the SUBST or ASSIGN commands, because DOS labels the "real" disk in the disk drive instead. Suppose that you have used the command **ASSIGN A=C**. If you then enter the command **LABEL A:**, DOS actually changes the volume label of the disk in drive C:.

Examples

1. **LABEL A:MYDISK**

   This command creates the volume label MYDISK on the diskette in drive A:. MYDISK replaces any previously existing volume label on the diskette.

2. **LABEL**

   The LABEL command requests a change in the volume label of the current disk drive (which, for this example, is drive A:). DOS responds with the following message:

```
Volume in drive A is MYDISK

Volume label (11 characters, ENTER for none)?_
```

   If you enter a valid volume label and press Enter, the old volume label MYDISK is replaced by the new label.

   If you only press Enter, DOS asks

```
Delete current volume label (Y/N)?
```

   If you press Y, DOS removes the current volume label. If you press N, LABEL exits and leaves the current label intact.

Notes

You can more easily identify your disks by giving each one a *volume label*, which is simply an electronic label for a disk. Volume labels appear when you use the DIR, TREE, or CHKDSK commands.

The volume label has one other use. When you format a hard disk with a volume label, FORMAT prompts you to type the label, thus confirming your intent. (If the disk has no volume label, FORMAT displays a warning message.) When you label a hard disk, you protect yourself from destroying megabytes of information with FORMAT.

When you format a disk with DOS V4, you are automatically prompted to enter a volume label. Whether or not you actually assign a label, DOS gives the disk a serial number. The serial number is not a part of the volume label. Remember that a space is a valid character in a volume label.

Spaces and underscores can increase the readability of a volume label. LABEL V3.0 and V3.1, however, reject a space in a volume name when the name is given on the command line (such as **LABEL MY DISK**). To put a space in a volume label, type **LABEL**, a space, and the disk drive name, if needed, and press Enter. Do not enter a volume label on the command line. When LABEL asks for a new volume label, you can enter the label with spaces.

Messages

1. `Cannot LABEL a network drive`

   ERROR: You tried to put a volume label on a networked disk drive. If this disk drive is "owned" by your computer, pause or disconnect from the network and reissue the LABEL command.

2. `Delete current volume label (Y/N)?`

   INFORMATION and WARNING: You did not enter a volume label when requested. DOS is asking if the current label should be deleted or left unaltered.

3. `Invalid characters in volume label`

   WARNING: You used a character that is not permitted in a volume label (probably a punctuation symbol like the period or comma).

4. `Invalid drive specification`

   ERROR: You named a nonexistent disk drive, named a disk drive that is part of a JOIN command, or omitted the colon in the disk drive name.

5. `No room in root directory`

   ERROR: The disk's root directory is full and cannot hold the new volume label. This error should not occur when you are changing or deleting a volume label, but only when you are adding a new volume label. Delete, or copy and delete, a file in the disk's root directory and reissue the command.

6. `Volume in drive d: is label_name`
   `Volume in drive d: has no label`

   INFORMATION: The floppy disk in drive `d:` has the volume label `label_name` if the first message appears. If the second message appears, the disk has no volume label.

MEM *V4—External*

Displays the amount of used and unused memory, allocated and open memory areas, and all programs currently in the system.

Syntax

*dc:pathc***MEM** */PROGRAM /DEBUG*

dc: is the optional drive that contains MEM.

pathc\ is the optional subdirectory where MEM is located.

Switches

/PROGRAM Displays programs that are in memory, including the address, name, size, and type of each file for every program. Also shows current free memory.

/DEBUG Displays programs that are in memory, including the address, name, size, and type of each file for every program. Also displays system device drivers and installed device drivers, as well as all unused memory.

Reference

See Chapter 8.

Rule

You cannot specify /PROGRAM and /DEBUG at the same time.

Examples

The following examples show some of the sample screen outputs that can result when you use MEM.

1. **MEM**

If you type **MEM** and press Enter, DOS responds with a message similar to the following:

```
655360 bytes total memory
655360 bytes available
581248 largest executable program size
```

2. **MEM /PROGRAM**

If you type **MEM /PROGRAM** and press Enter, DOS responds with a message similar to the following:

```
Address     Name          Size       Type
DDDDDDD     DDDDDDDD      DDDDDD     DDDDDD
000000                    000400     Interrupt Vector
000400                    000100     ROM Communication Area
000500                    000200     DOS Communication Area

000700      IO            002000     System Program
0027D0      MSDOS         008E20     System Program
00B5F0      IO            005330     System Data
             ANSI         001180      DEVICE=
                          000820      FILES=
                          000100      FCBS=
                          0029A0      BUFFERS=
                          0001C0      LASTDRIVE=
                          000CD0      STACKS=
```

```
010930      COMMAND      001640      Program
011F80      MSDOS        000030      -- Free --
011FC0      COMMAND      000130      Environment
012100      MEM          000060      Environment
012170      MEM          012F00      Program
025080      MSDOS        07AF70      -- Free --

        655360 bytes total memory
        655360 bytes available
        581248 largest executable program size
```

3. **MEM /DEBUG**

If, however, you type **MEM /DEBUG** and press Enter, DOS responds with a message similar to the following:

```
Address     Name          Size        Type
DDDDDDD     DDDDDDDD      DDDDDD      DDDDDD
000000                    000400      Interrupt Vector
000400                    000100      ROM Communication Area
000500                    000200      DOS Communication Area

000700      IO            002000      System Program
                CON                     System Device Driver
                AUX                     System Device Driver
                PRN                     System Device Driver
                CLOCK$                  System Device Driver
                A: - C:                 System Device Driver
                COM1                    System Device Driver
                LPT1                    System Device Driver
                LPT2                    System Device Driver
                LPT3                    System Device Driver
                COM2                    System Device Driver
                COM3                    System Device Driver
                COM4                    System Device Driver

0027D0      MSDOS        008E20      System Program

00B5F0      IO           005330      System Data
              ANSI       001180       DEVICE=
                         000820       FILES=
                         000100       FCBS=
                         0029A0       BUFFERS=
                         0001C0       LASTDRIVE=
                         000CD0       STACKS=
010930      COMMAND      001640      Program
011F80      MSDOS        000030      -- Free --
011FC0      COMMAND      000130      Environment
012100      MEM          000060      Environment
012170      MEM          012F00      Program
025080      MSDOS        07AF70      -- Free --
```

```
655360 bytes total memory
655360 bytes available
581248 largest executable program size
```

MKDIR or MD

V2, V3, V4—Internal

Creates a subdirectory.

Syntax

MKDIR *d:path***dirname**

MD *d:path***dirname**

d: is the disk drive for the subdirectory.

path\ indicates the path to the directory that will hold the subdirectory.

dirname is the subdirectory you are creating.

Reference

See Chapter 6.

Rules

1. If you do not specify a disk drive name, the current disk drive is used.
2. If you do not specify a path name, but give a disk drive name, the subdirectory is established in the current directory of the specified drive. If you do not specify a path name or a disk drive name, the subdirectory is established in the current directory of the current disk drive.
3. If you use a path name, separate it from the directory name with the path character, the backslash (\).
4. You must specify the new subdirectory name, which can have from one to eight characters, and an optional extension. The name must conform to the rules for creating directory names.
5. You cannot use a directory name that is identical to a file name in the parent directory. For example, if you have a file named MYFILE in the current directory, you cannot create the subdirectory MYFILE in this directory. If the file is named MYFILE.TXT, however, the names do not conflict, and you can create the MYFILE subdirectory.

Notes

MKDIR, or the short form MD, makes subdirectories. You can create as many subdirectories as you want, but remember: for the path name, DOS accepts no more than 63 characters, including backslashes. Do not create too many levels of subdirectories with long names, or the path to the directory may exceed 63 characters.

You are not restricted to creating subdirectories in the current directory. If you add a path name, DOS establishes a new subdirectory in whatever directory you specify.

You can create subdirectories on a disk used in an ASSIGN, JOIN, or SUBST command. Remember that the subdirectory is created on the physical disk drive used and not on the logical disk that DOS thinks is in use. The result is that the subdirectory actually resides on a disk drive different from the one you specified.

Message

`Unable to create directory`

ERROR: One of the following errors occurred: (1) you tried to create a directory that already exists; (2) you gave an incorrect path name; (3) the disk's root directory is full; (4) the disk itself is full; or (5) a file already exists by the same name.

Check the directory in which the new subdirectory was to be created. If a conflicting name exists, either change the file name or use a new subdirectory name. If the disk or the root directory is full, delete some files, create the subdirectory in a different directory, or use a different disk.

MODE—LPTPORT *V1, V2, V3, V4—External*

Sets the IBM printer characteristics.

Syntax

*dc:pathc***MODE LPTx**:*cpl,lpi, P*

If you have DOS V4.0 or V4.01, you can use the following form:

*dc:pathc***MODE LPTx**: **COLS**=*cpl* **LINES**=*lpi* **RETRY**=*ret*

dc: is the disk drive holding the command.

pathc\ is the path to the command.

x: is the printer number (1, 2, or 3). The colon is optional.

cpl is the number of characters per line (80, 132).

lpi is the number of lines per inch (six or eight).

P specifies continuous retries on time-out errors.

ret tells DOS what to do when a time-out error occurs. You can choose from the following options:

ret	*Action*
E	Return the error when port is busy (default).
B	Return busy when port is busy.
R	Return ready when port is busy (infinite retry).
NONE	Take no action.

Reference

See Chapter 9.

Rules

1. You must specify a printer number, but all other parameters are optional, including the colon after the printer number.
2. If you do not want to change a parameter, enter a comma for that parameter.
3. This command cancels the effect of **MODE LPTx: = COMy:**.
4. A parameter does not change if you skip it or enter an invalid parameter in its place. This is not true of the printer number, however, which must be entered correctly.
5. With versions through DOS V3.3, if you specify a *P* for continuous retries, you can cancel the *P* only by reentering the MODE command without the *P*. With DOS V4, retry option *B* has the same effect as the *P* option of earlier DOS versions.
6. If you are using a networked printer, you should not use the *P* option of DOS versions 3.3 and earlier, nor should you use the retry options of V4.
7. The characters-per-line and lines-per-inch portions of the command affect only IBM and Epson printers and other printers that use Epson-compatible control codes.

Notes

This command controls the IBM Matrix and Graphics Printers, all Epson printers, and Epson-compatible printers. The command may work partially or not at all on other printers.

When you change the column width, MODE sends the special printer-control code that specifies the normal font (80) or the condensed font (132). When you change the lines-per-inch setting, MODE sends the correct printer-control code for printing six or eight lines per inch. MODE also sets the printer to 88 lines per page for an eight lines-per-inch setting and to 66 lines per page for a six lines-per-inch setting.

If you give the option *P* in versions of DOS through DOS V3.3 or the *B* retry option of V4 and attempt to print on a deselected printer, the computer does not issue a time-out error. Rather, the computer internally loops until the printer is ready (turned on, connected to the PC, and selected). For about a minute, the computer appears to be locked up. To abort the continuous retry, press Ctrl-Break.

MODE—DISPLAY *V2, V3, V4—External*

Switches the active display adapter between the monochrome display and a graphics adapter/array (Color/Graphics Adapter, Enhanced Color/Graphics Adapter, or Video Graphics Array) on a two-display system, and sets the graphics adapter/array's characteristics.

Syntax

*dc:pathc***MODE dt**

*dc:pathc***MODE** *dt*, **s**, *T*

If you are using DOS V4.0 or V4.01, you can use the following form:

*dc:pathc***MODE CON**: **COLS**=*x* **LINES**=*y*

dc: is the name of the disk drive holding the command.

pathc\ is the path to the command.

dt is the display type, which may be one of the following:

40	Sets the display to 40 characters per line for the graphics display.
80	Sets the display to 80 characters per line for the graphics display.
BW40	Makes the graphics display the active display and sets the mode to 40 characters per line, black and white (color disabled).
BW80	Makes the graphics display the active display and sets the mode to 80 characters per line, black and white (color disabled).
CO40	Makes the graphics display the active display and sets the mode to 40 characters per line (color enabled).
C080	Makes the graphics display the active display and sets the mode to 80 characters per line (color enabled).
MONO	Makes the monochrome display the active display.

s shifts the graphics display right (**R**) or left (**L**) one character.

T requests alignment of the graphics display screen with a one-line test pattern.

x specifies the number of columns to display; 40 or 80 columns are possible.

y specifies the number of lines on the display. This is applicable for EGA and VGA displays only. EGA values are 25 and 80. VGA adds another value, 50.

Reference

See Chapter 9.

Rules

1. For the first form of the command, you must enter the display type (**dt**); all other parameters are optional.
2. For the second form of the command, you must enter the shift parameter **s** (which is an **R** or **L** for shifting right or left); the display type (**dt**) and test pattern (*T*) are optional.
3. The **s** (**R** or **L**) parameter works only with the Color/Graphics Adapter; the display does not shift if you use this command with any other adapter. The *T* parameter, conversely, will cause the test pattern to be displayed with any graphics adapter (Convertible, EGA, or VGA). If you use the parameter with the Monochrome Adapter, DOS responds with an error message.
4. When you use the shift and test pattern parameters, the screen is shifted, the test pattern is displayed, and DOS asks if the screen is aligned properly. If you answer Y for yes, the cursor returns to the system prompt (`A>`); by answering N for no, you cause the shift and prompting process to be repeated.
5. When DOS shifts the screen, the screen moves one character position left or right in 40-column mode and two character positions left or right in 80-column mode.
6. The *graphics display* is any display connected to the Color/Graphics Adapter (CGA), the Enhanced Graphics Adapter (EGA), PC Convertible, or Video Graphics Array (VGA). Either a color or monochrome video monitor can be connected to the adapter or array.
7. If your system does not have the display you specify, an `Invalid parameter` message appears.
8. Color is not displayed automatically when you use the CO40 or CO80 parameter for the display type. Programs that use color, however, can be displayed in color.
9. The display is cleared when you give any valid form of the command.

MODE—COM PORT *V1.1, V2, V3, V3.3, V4—External*

Controls the protocol characteristics of the Asynchronous Communications Adapter.

Syntax

*dc:pathc***MODE COMy**: **baud**, *parity, databits, stopbits, P*

If you are using DOS V4.0 or V4.01, you can use the following form:

*dc:pathc***MODE COMy**: **BAUD**=*baud PARITY*=*parity*
DATA=*databits STOP*=*stopbits RETRY*=*ret*

y: is the adapter number (1, 2, 3, or 4); the colon is optional after the number.

baud is the baud rate (110, 150, 300, 600, 1200, 2400, 4800, 9600, or 19200).

parity is the parity checking (None, Odd, or Even; in versions of DOS before V4.0, use N, O, or E).

databits is the number of data bits (7 or 8).

stopbits is the number of stop bits (1 or 2).

P represents continuous retries on time-out errors.

ret tells DOS what to do when a time-out error occurs. You can choose from the following options:

ret	*Action*
E	Return the error when port is busy (default).
B	Return busy when port is busy.
R	Return ready when port is busy (infinite retry).
NONE	Take no action.

Reference

See Chapter 9.

Rules

1. You must enter the adapter's number, followed by a space and a baud rate. If you type the optional colon, it must immediately follow the adapter number. All other parameters are optional.
2. If you do not want to change a parameter, enter a comma for that value.
3. If you enter an invalid parameter, DOS responds with an `Invalid parameter` message and takes no further action.
4. You can enter only the first two digits of the baud rate (for example, 11 for 110 baud and 96 for 9600 baud).

5. The 19200 baud rate is valid only for PS/2 computers. If you try to use the 19200 baud rate on a PC or compatible, DOS displays an `Invalid parameter` message and takes no further action.
6. If you want continuous retries after a time-out, you must enter the **P** or **RETRY=B** whenever you use the MODE COMn: command.
7. If the adapter is set for continuous retries (P or RETRY=B) and the device is not ready, the computer appears to be locked up. You can abort this loop by pressing Ctrl-Break.

MODE—REDIRECTION *V2, V3, V4—External*

Forces DOS to print to a serial printer rather than a parallel printer.

Syntax

*dc:pathc***MODE LPTx**: = **COMy**:

dc: is the disk drive holding the command.

pathc\ is the path to the command.

x: is the parallel printer number (1, 2, or 3). The colon is optional.

y: is the Asynchronous Communications Adapter number (1, 2, 3, or 4).

Reference

See Chapter 9.

Rules

1. You must give a valid number for both the parallel printer and the serial printer.
2. After you give the command, all printing that normally goes to the parallel printer goes to the designated serial printer.
3. This command can be canceled by the MODE LPTx: command.

Notes

This form of MODE is useful for systems that have only a serial printer. When you type the line

MODE LPT1: = COM1:

the serial printer receives all the output that usually is sent to the system printer (assuming that the serial printer is connected to the first Asynchronous Communications Adapter). This output includes the print-screen (Shift-PrtSc) function.

Before you issue the MODE LPT=COMy command, you should use the MODE COMn: command to set up the serial adapter used for the serial printer.

MODE—CODE PAGE PREP *V3.3, V4.0, V4.01—External*

Prepares (chooses) the code pages to be used with a device.

Syntax

*dc:pathc***MODE device CODEPAGE PREPARE = ((codepage**, *codepage, . . .*)*dp:pathp***pagefile.ext**)*

or

*dc:pathc***MODE device CP PREP** = (**(codepage**, *codepage, . . .*) *dc:pathp***pagefile.ext)**

dc: is the disk drive holding the command.

pathc\ is the path to the command.

device is the device for which code page(s) will be chosen. You can select one of the following devices:

CON:	The console
PRN:	The first parallel printer
LPTx:	The first, second, or third parallel printer (**x** is 1, 2, or 3)

codepage is the number of the code page(s) to be used with the device. The following are valid number(s):

437	United States
850	Multilingual
860	Portugal
863	French Canadian
865	Denmark/Norway

The ellipsis (. . .) represents additional code pages.

dp: is the disk drive holding the code page (font) information.

pathp\ is the path to the file holding the code page (font) information.

pagefile.*ext* is the file holding the code page (font) information. Currently, the DOS-provided code page files are as follows:

4201.CPI	The IBM ProPrinter
5202.CPI	The IBM Quietwriter III Printer
EGA.CPI	EGA/VGA type displays
LCD.CPI	IBM Convertible LCD display

Reference

See Chapter 15.

Rules

1. You must specify a valid device. The options are CON:, PRN:, LPT1:, LPT2:, or LPT3:. The colon after the device name is optional.
2. You must use the words **CODEPAGE** or **CP**, and **PREPARE** or **PREP**.
3. You must specify one or more code pages (**codepage**). You must separate each number with a comma if you give more than one code page. You must enclose the entire list of code pages in parentheses.
4. When you add or replace code pages, enter a comma for any code page that you do not want to change.
5. Do not use a hardware code page (the code page that is given to DISPLAY.SYS for the console or to PRINTER.SYS for the printers).
6. If you do not give a drive name, the current drive is used.
7. If you do not give a path name, the current directory is used.
8. You must specify the code page information file. Be sure to list the file's extension, if one exists.
9. A code page number is invalid if it does not appear in the code page information file. A file is invalid if the file exists but is not a code page information file. If an invalid code page number or file is used, DOS removes all current code pages in the same position as code pages given to the command.

Examples

1. **MODE CON CODEPAGE PREPARE = (850, 863, 865)**
   C:\EGA.CPI

   This command prepares the 850, 863, and 865 fonts for use with the EGA or VGA display. MODE gets its information on the fonts from the EGA.CPI file in the root directory of C:.

2. **MODE LPT1 CP PREP = (850, 863) C:\4201.CPI**

   This command prepares the 850 and 863 fonts for use with the IBM ProPrinter Model 4201, which should be connected to the first parallel port (LPT1). The information file is 4201.CPI. The command shows the abbreviations for CODEPAGE and PREPARE.

3. **MODE CON CP PREP = (,,437) C:\EGA.CPI**

   This example shows how to replace one code page while retaining previous code pages, by using a comma to mark the position where a code page should not be replaced. In this example, the 437 code page replaces whatever code page previously held the third position. If Example 1 were used

before this command, DOS would retain the 850 and 863 code pages but would replace the 865 code page with the 437 code page.

Notes

MODE CODEPAGE PREPARE is used to prepare code pages (fonts) for the console (keyboard and display) and the printers. You should issue this subcommand before issuing the MODE CODEPAGE SELECT subcommand, unless you are using the IBM Quietwriter III printer, whose font information is held in cartridges. If the needed code page is in a cartridge and the code page has been specified to the PRINTER.SYS driver, no PREPARE command is needed.

Example 3 shows how to replace some code pages but not others. When you prepare code pages, they are held in memory in the order you specify. When you give additional PREPARE commands, you replace code pages only in the same position.

MODE—CODE PAGE SELECT *V3.3, V4.0, V4.01—External*

Activates the code page used with a device.

Syntax

*dc:pathc***MODE device CODEPAGE SELECT = codepage**

or

*dc:pathc***MODE device CP SEL = codepage**

dc: is the disk drive holding the command.

pathc\ is the path to the command.

device is the device for which the code page(s) will be chosen. You can select one of the following devices:

CON:	The console
PRN:	The first parallel printer
LPTx:	The first, second, or third parallel printer (**x** is 1, 2, or 3)

codepage is the number of the code page(s) to be used with the device. The valid number(s) are as follows:

437	United States
850	Multilingual
860	Portugal
863	French Canadian
865	Denmark/Norway

Reference

See Chapter 15.

Rules

1. You must specify a valid device. The options are CON:, PRN:, LPT1:, LPT2:, or LPT3:. The colon is optional after the device name.
2. You must use the words **CODEPAGE** or **CP**, and **SELECT** or **SEL**.
3. You must specify a code page **codepage**. The code page must be part of a MODE CODEPAGE PREPARE command for the device, or it must be the hardware code page specified to the appropriate device driver.

Examples

1. **MODE CON CODEPAGE SELECT = 863**

   This command activates the 863 font for use with the EGA or VGA display. Thus the, 863 code page must be part of a previous MODE CODEPAGE PREPARE command or it must be the hardware code page for the EGA/VGA.

2. **MODE LPT1 CP SEL = 850**

   This command activates the 850 font for use with the printer attached to the first parallel port (LPT1:).

Notes

MODE CODEPAGE SELECT activates a currently prepared code page or reactivates a hardware code page. You can use MODE CODEPAGE SELECT only on these two types of code pages.

MODE CODEPAGE SELECT usually completes the downloading of any software font to the device, with the exception of the Quietwriter III printer, which uses cartridges. The Quietwriter III beeps when you select a code page, warning you to make sure that the proper cartridge is in the printer. DOS cannot tell which cartridge is in use. The wrong cartridge can yield nonsense characters.

MODE CODEPAGE SELECT activates code pages for individual devices. You can use the CHCP command to activate the code pages for all available devices.

For more information about CHCP and MODE SELECT, see the discussion on internationalizing DOS in Chapter 15. Also refer to CHCP in the Command Reference section of this book.

MODE—CODE PAGE REFRESH

V3.3, V4.0, V4.01—External

Reloads and reactivates the code page used with a device.

Syntax

*dc:pathc***MODE device CODEPAGE REFRESH**

or

*dc:pathc***MODE device CP REF**

dc: is the disk drive holding the command.

pathc\ is the path to the command.

device is the device for which the code page(s) will be chosen. You can select one of the following devices:

CON:	The console
PRN:	The first parallel printer
LPTx:	The first, second, or third parallel printer (**x** is 1, 2, or 3)

Reference

See Chapter 15.

Rules

1. You must specify a valid device. The options are CON:, PRN:, LPT1:, LPT2:, or LPT3:. The colon is optional after the device name.
2. You must use the words **CODEPAGE** or **CP**, and **REFRESH** or **REF**.

Notes

MODE CODEPAGE REFRESH downloads, if necessary, and reactivates the currently selected code page on a device. Use this command after you turn on your printer, or after a program alters the video display and leaves the console code page unusable.

You do not give a code page to MODE CODEPAGE REFRESH, because the command uses the last selected code page for the device.

MODE—CODE PAGE STATUS

V3.3, V4.0, V4.01—External

Displays a device's code page status.

Syntax

*dc:pathc***MODE device CODEPAGE** */STATUS*

*dc:pathc***MODE device CP** */STA*

dc: is the disk drive holding the command.

pathc\ is the path to the command.

device is one of the following:

CON:	The console
PRN:	The first parallel printer
LPTx:	The first, second, or third parallel printer (**x** is 1, 2, or 3)

Switch

/STATUS, or its abbreviated form, */STA*, displays the status of the device's code pages.

Reference

See Chapter 15.

Rules

1. You must specify a valid device. The options are CON:, PRN:, LPT1:, LPT2:, or LPT3:. The colon is optional after the device name.
2. You must use the word **CODEPAGE** or **CP**.
3. You can use the /STATUS or /STA switch. If you omit all other MODE keywords and the switch (that is, if you type only **MODE** and press Enter), MODE assumes that you want the STATUS subcommand.
4. MODE /STATUS displays the following information about the device:
   a. The selected (active) code page, if one has been selected.
   b. The hardware code page(s).
   c. Any prepared code page(s).
   d. Any available positions for additional prepared code pages.

Examples

1. **MODE CON CODEPAGE /STATUS**

   This command displays the console's code page status.

2. **MODE LPT1 CP**

This command displays the code page status of the printer attached to the first parallel port (LPT1:). The /STATUS switch is optional. If you give only the device name and the word **CODEPAGE** (or **CP**) in the command line, MODE assumes that you want to see the device's status.

Notes

MODE CODEPAGE /STATUS shows the device's current code page status.

Here is a sample output for the command:

```
Active codepage for device CON is 437
hardware codepages:
  Codepage 437
prepared codepages:
     Codepage 850
     Codepage not prepared
MODE Status Codepage function completed
```

The first line displays the active code page for the device. If no code page has been selected, the command instead displays

```
No codepage has been SELECTED
```

and the device uses the hardware code page. The hardware code page is used until a different code page is selected by means of the MODE CODEPAGE SELECT or CHCP commands.

The next lines indicate the hardware code page given to the DISPLAY.SYS driver for the console, or to the PRINTER.SYS driver for the printers. DOS usually shows only one hardware code page but two code pages may be shown for the IBM Quietwriter III printer.

The message then lists the code pages available (prepared) for the device. Every code page is displayed, one per line. Lines that say `Codepage not prepared` mean that positions are available for additional code pages, but no code pages occupy these slots. You should see the number of additional code pages a device can use; this number is given to the DISPLAY.SYS or PRINTER.SYS driver.

MODE—KEY REPEAT *V4—External*

Adjusts the rate at which the keyboard repeats a character.

Syntax

*dc:pathc***MODE CON RATE**=*x* **DELAY**=*y*

dc: is the drive containing the MODE command.

pathc\ is the optional path containing the MODE command.

x is a value that specifies the character-repeat rate. You can select a value between 1 and 32.

y is a value that specifies the length of delay between the initial pressing of the key and the start of automatic character repetition. This value can be 1, 2, 3, or 4, which represent delays of 1/4 second, 1/2 second, 3/4 second, and one full second, respectively.

Reference

See Chapter 9.

MODE—STATUS *V4—External*

Displays the status of a specified device or of all devices that can be set by MODE.

Syntax

*dc:pathc***MODE device /STA***TUS**

dc: is the disk drive holding the command.

pathc\ is the path to the command.

device is the optional device to be checked by MODE.

Switch

/STA*TUS* is the mandatory switch to check the status of a device or devices. If you prefer, you can enter just **/STA** rather than the complete **/STATUS**.

Reference

See Chapter 9.

Note

This command enables you to see the status of any device that you normally set with MODE. Typing **MODE LPT1 /STA**, for example, displays the status of the first parallel port.

MODE—GENERAL *V1, V2, V3, V4—External*

Sets the mode of operation for the printer(s), the video display, and the Asynchronous Communications Adapter. Also controls code page switching for the console and printer. Other functions differ according to DOS implementation. See appendixes for machine variations.

Reference

See Chapter 8.

Notes

The MODE command is actually several commands in one. MODE is used to set and control the characteristics for the printer, the video display and the active display (in systems with more than one display), the Asynchronous Communications Adapter, and the keyboard rate (DOS V4). The command also enables you to set and control the redirection of printing between parallel and serial printers.

Starting with DOS V3.3, the MODE command also concerns code pages (font files) for international language use of your console printer. These MODE commands set and control the following functions:

MODE COMMAND	*FUNCTION*
PREPARE	Chooses code pages for use
SELECT	Activates code pages
REFRESH	Reestablishes lost code page information
/STATUS	Displays current code page

Messages

1. `Active code page for device devicename is codepage`

   INFORMATION: The MODE /STATUS command displays the number of the active code page (`codepage`) for the device (`devicename`).

2. `Codepage not prepared`

   INFORMATION or ERROR: If you used the MODE /STATUS command, DOS is telling you that a space is available for a prepared code page, but no code page has been prepared for this space.

   If you used the MODE SELECT command, you selected a code page that is not prepared. If you issue the command for the console device (CON:), the selected code page may not have the correct font for the current video mode. MODE SELECT aborts.

   Use MODE PREPARE to reload the desired code page, then reissue the MODE SELECT command. If the error recurs, edit the DEVICE = DISPLAY.SYS line to increase the number of *subfonts* and restart DOS.

   This message may precede other MODE error messages.

3. `Codepage operation not supported on this device`

   ERROR: If you used either the MODE PREPARE or MODE SELECT command, one of the following occurred: (1) you specified a nonsense device;

(2) you misspelled the device name; (3) you specified a device that does not exist on your system; (4) you named a device that does not have the proper device driver installed (DISPLAY.SYS for the console or PRINTER.SYS for the printer); (5) you incorrectly phrased the directive for the device driver; or (6) you specified a device that cannot switch code pages.

First check the spelling of the device you specified to the MODE command. If the device's name is correct, make sure that you have the right phrasing for the device driver in CONFIG.SYS. Check the CONFIG.SYS file for the DISPLAY.SYS line (if the problem is in the CON device) or the PRINTER.SYS line (if the problem is in the LPTx or PRN devices). Make sure that you have specified the correct LPT number for your printer. If you change the CONFIG.SYS file, restart DOS and try the command again.

4. `Code pages cannot be prepared`

   ERROR: When you used the MODE PREPARE command, you may have specified more code pages than the device driver allows. Otherwise, you may have duplicated a code page for the Quietwriter III printer. In response, MODE does not prepare any code pages. Further, MODE deletes the previously prepared code pages that are in the same position as this command's code pages.

   If you use the Quietwriter III, you cannot duplicate a hardware code page. If you use any other device, use the MODE /STATUS command to check the number of prepared code pages allowed. If you need more code pages than are listed in the prepared code page lines, edit the *added_codepages* parameter in the DISPLAY.SYS or PRINTER.SYS line of your CONFIG.SYS file. Restart DOS and try the command again.

5. `COMy bbbb,p,d,s,t initialized`

   INFORMATION: The Asynchronous Communications Adapter was initialized successfully. `y` is the adapter number, `bbbb` is the baud rate, `p` is the parity, `d` is the number of data bits, `s` is the number of stop bits, and `t` is the retry on time-out.

6. `Current keyboard does not support this Codepage`

   WARNING: You already issued the KEYB command. You issued a MODE SELECT command, and a conflict exists between the keyboard's current code page and the console code page you just specified to MODE. The console code page is set to the specified code page, but the keyboard code page is not changed. Nonsense characters appear when you type on the keyboard.

   To use this new code page on the console, you must issue the KEYB command with the correct, compatible keyboard code. KEYB automatically switches the keyboard's code page.

7. `Device error during Prepare`
   `Device error during Refresh`
   `Device error during Select`
   `Device error during Status`
   `Device error during write of font file to device`

   ERROR: DOS detected a device error during a MODE CODEPAGE operation. The error has occurred for one of the following reasons: (1) the device does not support code page switching; (2) the CONFIG.SYS file does not have a line for the proper device driver; (3) the number of *added_codepages* is too low to allow for additional code pages; (4) the code page information file is ruined; (5) you forgot to give the code page information file's full name in a MODE PREPARE command; (6) you specified an incorrect code page number; (7) the device detected a transmission error; (8) the device may be turned off or not selected; or (9) you used the MODE REFRESH command before you gave a MODE SELECT or CHCP command.

   If you used MODE REFRESH, issue the MODE SELECT command instead. Make sure that you have given the proper device name, code page number, and the code page file's full name, and that the device is turned on and ready. Then try the command again.

   If DOS responds with the same error message again, check the CONFIG.SYS file to make sure that the proper device driver (DISPLAY.SYS for the console or PRINTER.SYS for the printer) has been included and that the proper parameters have been given. If the line is there and is properly phrased, check the number of additional code pages. If this number is too low, edit the line and increase the number of added code pages. Restart DOS and try the command again.

   If the message reappears, you probably have a hardware problem. Make sure that the printer connections are secure and that the printer is receiving characters. Try Shift-PrtScrn. If nothing prints, your printer has a problem. If the screen's contents print and the MODE command still does not work, your printer has an internal electronics malfunction.

8. `Device or codepage missing from font file`

   ERROR: When you used the MODE PREPARE command you specified a correct code page, but the code page information file does not contain the font. You may have tried either to prepare a hardware code page or to use a font that the device does not support. All existing code pages for this device are discarded and must be prepared again.

   Try the command again and use another correct code page number. If this works, the device can use code pages, but you cannot use the code page that you originally specified. Also, be sure that you have specified the correct hardware code page to the appropriate device driver. If the hardware code page is incorrect, edit the CONFIG.SYS file, restart DOS, and try the command again.

9. `Do you see the leftmost ▯? (Y/N)`
   `Do you see the rightmost ▯? (Y/N)`

   INFORMATION: You are adjusting the display connected to the Color/Graphics Adapter. Answer N or press any other key to shift the display left or right (depending on how you invoked MODE). Answer Y if the display is centered properly. This answer ends the MODE command.

10. `Error during read of font file`

    ERROR: When you gave the MODE PREPARE command, DOS read the code page information file and detected a disk error, which probably was caused by a bad sector on the disk. MODE aborts and does not change the prepared fonts.

    Run the RECOVER command on the file. If RECOVER reports any bad sectors on a floppy disk, reformat or retire the floppy disk. If the problem is found on a hard disk, RECOVER hides the disk's bad sectors and the code page information file is ruined. Copy the .CPI file from your start-up floppy disk to the hard disk, and try the command again.

11. `Failure to access Codepage Font File`

    ERROR: You used the MODE PREPARE command, but MODE did not find the code page information file you specified. You omitted a drive or path name, misspelled the file name, or gave a nonsense file name. Be sure that the code page information file name is spelled correctly and that it is on the correct disk and directory.

12. `Failure to access device: devicename`

    ERROR: MODE CODEPAGE could not open the device `devicename`. You gave the wrong device name, or the proper device driver is not installed in CONFIG.SYS. If this is not the case, then the device itself may be turned off or disconnected.

    If you gave the wrong device name, reissue the command with the correct device name. If the device is a printer, make sure that the printer is connected and turned on. You also should make sure that the proper device driver (DISPLAY.SYS for the console or PRINTER.SYS for the printer) is loaded via CONFIG.SYS and phrased properly. If you find an error in CONFIG.SYS, edit the file, restart DOS, and try the command again.

13. `Font file contents invalid`

    ERROR: During a MODE PREPARE command, DOS found the file you specified as the code page information file, but the file's contents are incorrect. Either you specified an existing file that is not a code page information file, or the code page information file has been ruined. All the device's existing code pages are discarded.

    Be sure that you have given MODE the right file name, and reissue the command. If DOS again responds with the error message, the code page file

is damaged. Copy the code page information file from your start-up floppy disk, and try the command again.

14. `Illegal Device Named`

    ERROR: One of the following errors occurred: (1) you did not specify a number with LPTx: or COMy:; (2) you specified a number that is not in the correct range (1, 2, or 3 for LPTx: or 1 or 2 for COMy: when COMy: is used with a PC, or 1, 2, 3, or 4 for PS/2 computers); (3) you put no space between LPTx: or COMy: and the next parameter; (4) you put more than one space between LPTx: or COMx: and the next parameter; or (5) the specified adapter is not connected to your system.

15. `Infinite retry on parallel printer timeout`
    `No retry on parallel printer timeout`

    INFORMATION: You used the MODE LPTx or COM commands. The first message appears if you specify the *,P* option. The second message appears if you do not specify the *,P* option.

16. `Infinite retry not supported on Network printer`

    ERROR: You used the MODE LPTx command's *,P* option, and the printer you specified is owned by another computer or is part of a network. You cannot use *,P* on this printer. Either give the command again without the *,P* or specify a printer that is owned by your computer and not used by a network.

17. `Invalid baud rate specified`

    ERROR: You gave an incorrect baud rate. The baud rate can be 110, 150, 300, 600, 1200, 4800, 9600, or 19200 (or the first two characters for one of these numbers).

18. `Invalid parameter 'string'`

    ERROR: MODE detected a phrasing error in a code page command. The objectionable word is `string`, which you may have abbreviated to three or four letters. Otherwise, you (1) forgot to insert the equal sign or to surround the code pages with parentheses in a MODE PREPARE command; (2) reversed the word order; (3) forgot to give the word CODEPAGE (or CP); (4) forgot to name the device; or (5) used a word that MODE does not recognize.

19. `Invalid parameters`

    ERROR: One of the following errors occurred: (1) you forgot to give a necessary parameter; (2) the first parameter (for the printer or communications adapter) did not start with *L* or *C*; (3) you used an incorrect parameter in the command; or (4) you specified a display adapter that is not connected to your system. Check your spelling and punctuation and reissue the command.

20. `LPTx: not rerouted`

    INFORMATION: The specified printer adapter is receiving its normal data. Any previous reassignment (LPTx: = COMy:) has been canceled. This message appears when you set or adjust the characters per line or the lines per inch for the printer.

21. `LPTx: rerouted to COMy:`

    INFORMATION: The specified communications adapter is now receiving the output that usually goes to the specified printer adapter.

22. `LPTx: set for 80`
    `LPTx: set for 132`

    INFORMATION: The specified printer is set for 80 or 132 characters per line.

23. `MODE action Codepage function completed`

    INFORMATION: MODE has completed its function, which is shown as `action` in this generic form of the message. In an actual message, `action` is replaced by the function that has been completed—`Prepare`, `Refresh`, `Select`, or `Status`. This message appears after a MODE CODEPAGE function has run successfully.

24. `Must specify COM1, COM2, COM3 or COM4`

    ERROR: You gave an invalid number for COMy in a MODE LPTx=COMy command. Check the number you gave and try again.

25. `No codepage has been SELECTED`

    INFORMATION: You tried to use the MODE /STATUS command, but you have not yet used the MODE SELECT command on this device.

26. `Printer error`

    ERROR: You used the MODE LPTx command, and MODE could not set the printer. The printer may be turned off, not selected, not connected, out of paper, or not IBM-compatible. Check your printer and try the command again.

27. `Printer lines per inch set`

    INFORMATION: You specified a parameter for the lines per inch (six or eight). If your attempt to set the lines per inch had failed, a `Printer error` message may have followed this message.

28. `Resident portion of MODE loaded`

    INFORMATION: A section of MODE has been loaded and has become a resident part of DOS. This message appears when you use the continuous retry option for MODE LPT#: or MODE COMx:, when you shift the display left or right, or when you reroute printers (MODE LPT#: = COMx). MODE stays resident until you restart DOS.

29. `Unable to shift screen left`
    `Unable to shift screen right`

    ERROR: You used the MODE display command and asked DOS to shift the display for a Monochrome Adapter, an Enhanced Color/Graphics Adapter, or a Video Graphics Display. You cannot shift a display that is connected to any of these adapters.

MORE *V2, V3, V4—External*

Purpose

Displays one screen of information from the standard input device and pauses while displaying the message `'More'`.

Syntax

*dc:pathc***MORE**

dc: is the disk drive holding the command.

pathc\ is the path to the command.

Reference

See Chapter 15.

Rules

1. MORE displays one screen of information on the standard output (display).
2. After MORE displays a screen of information, it waits for a keystroke before filling the screen with new information. This process is repeated until all output has been displayed.
3. MORE is useful with I/O redirection and piping.

Examples

1. **MORE <TEST.TXT**

   MORE displays a screenful of information from the file TEST.TXT, then displays the prompt `--More--` at the bottom of the screen and waits for a keystroke. When you press any key, MORE continues this process until all information has been displayed. The prompt does not appear on the final screenful of information.

2. **DIR | SORT | MORE**

   MORE displays, 23 lines at a time, the sorted output of the directory command.

Notes

MORE is a DOS filter that enables you to display information without manually pausing the screen.

MORE is similar to the TYPE command, but MORE automatically pauses after each screen of information.

One screen of information is based on 40 or 80 characters per line and 23 lines per screen. MORE, however, does not always display 23 lines from the file. Rather, the command acts intelligently with long lines, wrapping those that exceed the display width (40 or 80 characters). If one of the file's lines takes three lines to display, MORE displays a maximum of 21 lines from the file, pauses, and shows the next screen.

NLSFUNC *V3.3, V4—External*

Supports country information; allows the CHCP command.

Syntax

*dc:pathc***NLSFUNC** *d:path***filename**.*ext*

dc: is the disk drive holding the command.

pathc\ is the path to the command.

d: is the disk drive holding the country information file.

path\ is the path to the country information file.

filename.*ext* is the country information file. With DOS V3.3, this information is contained in the file COUNTRY.SYS.

Reference

See Chapter 15.

Rules

1. If you do not name a drive for the country information file, the current disk drive is used.
2. If you do not give a path name, the current directory is used.
3. If you give a drive or path name, you also must give the name of the information file, which usually is COUNTRY.SYS.
4. If you omit the full file name, DOS searches for the file COUNTRY.SYS in the current disk's root directory.
5. After NLSFUNC is loaded, it remains active until you restart DOS.

Examples

1. **NLSFUNC**

   This command loads the NLSFUNC program. NLSFUNC automatically searches for the COUNTRY.SYS file in the current disk drive's root directory.

2. **NLSFUNC C:\BIN\COUNTRY.SYS**

   This command loads the NLSFUNC program and directs NLSFUNC to find COUNTRY.SYS in the \BIN directory of drive C:.

Notes

NLSFUNC loads support for extended country information and enables you to use the CHCP (Change Code Page) command. When you load NLSFUNC, DOS increases in size by 2,672 bytes.

NLSFUNC contains the code used for DOS function number 66 hexadecimal (base 16). The function call provides information about the conventions for the date, time, numeric and currency formatting, upper- and lowercase translations, and the character sorting order for the current country code. The current country code is defined by your system's default COUNTRY code, or by the COUNTRY directive used in CONFIG.SYS.

NLSFUNC also supports the CHCP command. NLSFUNC must be loaded before CHCP can be used.

You must load NLSFUNC only if your programs use the extended country information that is available from DOS, or if you want to use the CHCP command to change code pages. Your program's documentation should tell you if NLSFUNC is required.

If you use DOS V3.3, however, you may encounter a problem when you issue the NLSFUNC command. According to the documentation for V3.3, if you omit the COUNTRY.SYS file's full file name, NLSFUNC uses the name supplied to the COUNTRY directive in CONFIG.SYS. Unfortunately, DOS seems to forget the COUNTRY.SYS file's location when you omit the path. For this reason, you should either give the full file name for COUNTRY.SYS to NLSFUNC or place COUNTRY.SYS in the root directory of your start-up disk drive. Either procedure will enable NLSFUNC to find the COUNTRY.SYS file. This problem has been resolved in DOS V4.

You can use INSTALL in DOS V4 to activate the NLSFUNC command from CONFIG.SYS. After specifying the COUNTRY line in CONFIG.SYS, use the following form:

INSTALL=*dc:path***NLSFUNC** *d:path**filename.ext*

This keeps you from starting NLSFUNC at the DOS prompt.

Messages

1. `File not found`

   ERROR: NLSFUNC could not locate the country information file, COUNTRY.SYS. You may have forgotten to specify the disk drive, path, or file name for COUNTRY.SYS. Otherwise, you omitted the full file name and COUNTRY.SYS is not in the root directory. You should either copy COUNTRY.SYS to the root directory or give the full path and file name for COUNTRY.SYS.

2. `NLSFUNC already installed`

   WARNING: NLSFUNC has already been installed. You should use NLSFUNC only once after DOS has started.

PATH *V2, V3, V4—Internal*

Tells DOS to search specific directories on the specified drives if a program or batch file is not found in the current directory.

Syntax

PATH *d1:path1;d2:path2;d3:path3;...*

d1:, *d2:*, and *d3:* are valid disk drive names.

path1, *path2*, and *path3* are valid path names to the commands you want to run while in any directory.

The ellipsis (...) represents additional disk drives and path names.

Reference

See Chapter 6.

Rules

1. If you do not enter a disk drive name for a path, the current disk drive is used.
2. A path can be any series of valid subdirectories, separated by backslashes.
3. If you specify more than one set of paths, the following rules apply:
   a. The path sets must be separated by semicolons.
   b. The search for the programs or batch files is made in the order in which you give the path sets. First, the current directory is searched. Then *d1:path1* is searched, then *d2:path2*, *d3:path3*, and so on, until the command or batch file is found.
4. If you give invalid information when assigning a path to search (such as an invalid disk drive, a bad delimiter, or a deleted path), DOS does not display an error message. When the path is being searched for a program or batch file, DOS skips a bad or invalid path. If, however, you give an invalid disk drive name, DOS displays an `Invalid drive in search path` message when it searches the path for a program or batch file.
5. PATH affects the execution of only program (.COM or .EXE) files or batch (.BAT) files. This command does not work with data files, text files or program overlays.

Notes

PATH is a useful command in hierarchical directories. Directories containing utility programs, system programs, or batch files can be established and used from anywhere on the disk. PATH automatically causes DOS to search these directories to find your programs; you do not have to type the path each time.

Remember that the PATH command locates only program files and batch files. Once it has been invoked, your program—not DOS—must search other directories for data files, text files, or program overlays. Programs like WordStar Release 3 (not WordStar Release 4 or 2000) must have their overlay files in the directory you are using. Future versions of programs will enable you to "install" a path. The DOS V3.3 and V4 APPEND command, however, can be used to trick programs into finding their overlays or data files in different directories.

To view the current path, use either the PATH or the SET command with no arguments.

Always specify an absolute path that starts with a disk's root directory (\) rather than a relative path that starts with the current directory.

By specifying an absolute path, you do not have to be concerned about the current directory. For example, suppose that you keep your computer's DOS programs in a level-two directory called BIN and in a level-three directory called UTIL. You can set the path as

```
PATH C:\BIN;C:\BIN\UTIL
```

This command enables you to invoke a DOS command from anywhere on any disk, whether it is a hard, floppy, or RAM disk. If, on the other hand, you use the PATH command and a relative path starting with the current or parent directory, you must change the path almost every time you change subdirectories.

Because you also have started each path name with a disk drive name, DOS always searches the subdirectories \BIN and \BIN\UTIL on drive C:, regardless of which disk drive is current. You can be working on any disk drive, not just the hard disk, and invoke the DOS commands from BIN and UTIL.

When you type the name of a program or batch file, the current directory is searched. If the program or batch file is not found, DOS searches through each path in sequence. If the program or batch file is not found in any of the paths, DOS displays the `Bad command or filename` error message.

Message

`Invalid drive in search path`

WARNING: You specified a nonexistent disk drive name in one of the paths. This message appears when DOS searches for a program or batch file, not when you give the PATH command.

Use PATH or SET to see the current path. If the disk drive is temporarily invalid because of a JOIN or SUBST command, you can ignore this message. If you have

specified the wrong disk drive, issue the PATH command again and give the complete set of directory paths you want to use.

PRINT *V2, V3, V4—External*

Causes the printer to print a list of files while the computer performs other tasks.

Syntax

*dc:pathc***PRINT** */D:device /B:bufsiz /M:maxtick /Q:maxfiles /S:timeslice /U:busytick d1:path1**filename1.ext1 /P/T/C d2:path2**filename2.ext2/P/T/C ...*

d1: and *d2:* are valid disk drive names.

path1\ and *path2*\ are valid path names to the files for printing.

filename1.ext1 and *filename2.ext2* are the files you want to print. Wild cards are allowed.

The ellipsis (...) represents additional file names in the form *dx:pathx**filenamex.extx*.

Switches

You can specify any one of the following switches, but only the first time you start PRINT:

/D:device Specifies the device to be used for printing. *device* is any valid DOS device name. (You must list this switch first whenever you use it.)

/B:bufsiz Specifies the size of the memory buffer to be used while the files are printing. *bufsiz* can be any number from 1 to 16,386.

/M:maxtick Specifies in clock ticks the maximum amount of time that PRINT has for sending characters to the printer every time PRINT gets a turn. (A tick is the smallest measure of time used on PCs. A tick happens every 1/18.2 [0.0549] seconds.) *maxtick* can be any number from 1 to 255. Default is 2.

/Q:maxfiles Specifies the number of files that can be in the queue (line) for printing. *maxfiles* can be any number from 4 to 32. Default is 10.

/S:timeslice Specifies the number of slices in each second. *timeslice* can be a number from 1 to 255. Default is 10.

/U:busytick Specifies in clock ticks the maximum amount of time for the program to wait for a busy or unavailable printer. *busytick* can be any number from 1 to 255.

You can specify any one of the following switches whenever you use PRINT:

/P Queues up the file(s) (*places* the file(s) in the line) for printing.

/T *Terminates* the background printing of any or all files, including any file currently being printed.

/C *Cancels* the background printing of the file(s).

Reference

See Chapter 9.

Rules

1. If you do not name a disk drive, the current disk drive is used.
2. If you do not give a path name, the current directory is used.
3. If you do not give a file name, the background printing status is displayed.
4. You can specify any of the switches from the first set (/D, /B, /M, /Q, /S, and /U) only when PRINT is first used. If you give the /D switch, you must make it the first switch on the line. You can give the group's remaining five switches in any order before you specify a file name.
5. The /D switch specifies the device to which you will print. Unless you have a serial printer, PRN works well. The remaining switches' uses are less obvious. If you do not give the /D switch the first time you use PRINT, DOS displays the following prompt:

```
Name of list device [PRN]:
```

You should respond to the prompt in one of two ways:

a. Press Enter to send the files to PRN (normally LPT1:). If LPT1: is redirected (see the MODE command), the files are rerouted.

b. Enter a valid DOS device name. Printing will be directed to this device. If you enter a device that is not connected to your system, PRINT will accept files in the queue. The files will not be processed, however, and you will lose processing speed.

You cannot change the assignment for background printing until you restart DOS.

6. If you do not give any of the other switches from the first set, DOS assumes the following default values for those switches:

Switch	*Default Value*
bufsiz	512 bytes
maxtick	2 ticks
maxfiles	10 files
timeslice	8 slices/second
busytick	1 tick

7. When you give a switch from the second set (/P, /T, and /C), the switch affects not only the file name preceding it, but all succeeding files, as well, until the next switch in the line.
8. If you name a file without a switch, DOS assumes that the /P (print) switch has been given.
9. Do not try to stuff a queue with more files than you have specified in the *maxfiles* switch. (If the /Q switch was not given, DOS assumes that the queue will hold a maximum of ten files.) If you try to violate the *maxfiles* setting, PRINT responds with the message `PRINT queue is full` and ignores the command.
10. Files are printed in the order in which you give their names. If wild cards are used, the files are printed in the order in which they are listed in the directory.
11. If DOS detects a disk error in a file that PRINT is trying to print, DOS cancels the printing of that file, displays an error message, and prints the next files in line.
12. The command PRINT /C has no effect if no file name is specified.
13. To print on a networked printer, you must use the NET PRINT command rather than PRINT.
14. The first time you invoke PRINT, DOS increases in size by approximately 5,500 bytes. When you increase or decrease certain default settings, you proportionally change the size of DOS. To regain this memory space, however, you must restart DOS.

Notes

PRINT controls the background printing feature of DOS V3 and V4. The command has been significantly improved since DOS V2.

During the background printing process, DOS orders a specified printer to print a disk-based file while the computer itself performs a separate task (such as running a program). Because the computer must devote most of its time and attention to the nonprinting task, that task is "in the foreground." PRINT gets less attention from the computer, and therefore is called a "background" task. When the cursor sits at the system prompt (`A>`), PRINT gets almost all the computer's time.

The /B switch acts like a disk buffer. PRINT reads into memory a portion of the document to print. As you increase the value of *bufsiz*, you decrease the number of times that PRINT must read the file from the disk, and thereby increase printing throughput. Always use a multiple of 512, such as 1,024, 2,048, and so on, as the value of *bufsiz*. The default size (512 bytes) is adequate for most uses, but using **/B:4096** increases performance for most documents of two pages or less.

For the /M or /S switch, the following formula shows how much CPU time PRINT gets per second:

$$\text{PRINT's \% of CPU time} = \frac{maxtick}{(1 + timeslice)} * 100$$

When the default values are assumed, PRINT gets 22 percent of the computer's time. Increasing *maxtick* gives PRINT more time; increasing *timeslice* gives PRINT less time. Because the keyboard action becomes sluggish as PRINT uses more time, the default values for PRINT usually work well.

The /U switch tells PRINT how long to wait for a busy printer. The switch's default value is 1, but you can raise this number to between 2 and 8 if you use a printer that works at less than 4800 baud (a serial printer) or less than 100 cps (a parallel printer).

The word *queue* comes from the Old French word for "tail" or "line." When you invoke the PRINT command to print a file, you "queue" the file, or add it to a line of files that are waiting to be printed. DOS prints these files as free time becomes available.

Background printing is best used with program files stored in ASCII format, but the PRINT command prints any disk file. All the file's characters, including control characters, are transmitted to the printer. An exception is made for the tab character Ctrl-I, or CHR$(9), however, which DOS pads with spaces to the next eight-column boundary. If you attempt to background-print a .COM or an .EXE file, you may cause your printer to eject paper, beep, change print styles, or do other surprising things.

The positioning of the /P, /C, and /T switches on the command line is important. Each switch affects the file immediately preceding it, as well as all the following files, until another switch is encountered. For example, the command

PRINT LETTER.TXT /P PROGRAM.DOC MYFILE.TXT /C TEST.DOC

places the files LETTER.TXT and PROGRAM.DOC in the queue to be background-printed. The /C switch, however, cancels the background printing of MYFILE.TXT and TEST.DOC.

In this example, the /P switch affects the preceding file (LETTER.TXT) and the following file (PROGRAM.DOC). Similarly, the /C switch affects the preceding file (MYFILE.TXT) and the following file (TEXT.DOC).

If you use the /T switch, background printing is canceled for all files in the queue, including the file currently being printed. You do not need to give the /T switch with a file name because the switch terminates printing for all files, including any file names given on the command line.

If you cancel or terminate background printing while a file is being printed, a cancellation message is printed on the printout. If you terminate all files, the message says "All files canceled by operator." If you cancel the file currently being printed (by using the /C switch), the file name is printed along with the message "File canceled by operator." The printer then form feeds, and the printer's bell rings.

If you do not cancel all files, printing continues with the next queued file.

If a disk error occurs during background printing, DOS cancels the current file and places a disk-error message on the printout. The printer then performs a form feed, the bell rings, and DOS prints any remaining files in the queue.

When PRINT has control of the printer, you should not try to print anything else. If you try to print while using a program that is written in BASIC, DOS responds with an `out of paper` error message. If you are at the system level, DOS displays a `Device not ready` error message.

If either message appears, you should press **A** for abort and try later. If you attempt to print in the middle of a program, DOS suspends (hangs up) the program or displays an error message.

When you print files on a floppy disk, do not remove the floppy disk until PRINT finishes printing the files.

You should never edit, alter, or erase any file being printed with the PRINT command. DOS may not print the correct copy of the file if it has been edited or altered. Further, DOS skips the file if the floppy disk is removed or if the file is deleted (as though DOS had a disk error).

In DOS V3 and V4, PRINT sensibly handles file queuing if you give more than ten files without adjusting the *maxfiles* switch, or if you violate the number of files set by *maxfiles*. PRINT V3 and V4 respond with an error message and continue to work normally. PRINT V2, on the other hand, acts erratically. If you queue more than ten files, you lose control over PRINT V2 until it finishes printing.

Messages

1. `Access denied`

   ERROR: You attempted to print a file that is for the exclusive use of another computer or program. Try the command later.

2. `All files canceled by operator`

   INFORMATION: You used the /T switch to terminate all queued files.

3. `Cannot use PRINT – Use NET PRINT`

   ERROR: You attempted to print a file to a networked printer, either on your own computer or on another computer. You must use NET PRINT, not PRINT, to print to this printer. You may pause or disconnect from the network and use PRINT on your own printer.

4. `Errors on list device indicate that it may be off-line. Please check it.`

   WARNING: The PRINT device probably is not selected, connected, or turned on. Check your printer and cable, and make sure that all connections are correct.

5. `File filename canceled by operator`

   INFORMATION: This message appears on the printout to remind you that the printout of the file `filename` is incomplete because you canceled the printing.

6. `filename File not found`

   WARNING: PRINT did not find the specified file, `filename`. When this happens, PRINT skips the file and processes any remaining files that are named on the command line. You should look for this file in the disk drive, path, and file names. Verify the file's location and try the command again.

7. `filename File not in PRINT queue`

   WARNING: You used the /C switch to cancel a file (`filename`) that was not in the PRINT queue. Either you already have printed the file or you have misspelled its name. If you meant to cancel one or more files and print others (by giving both the /C and /P switches), either you gave the /C switch out of order or you omitted the /P switch. Check each of these possibilities and reissue the command.

8. `Invalid drive specification`

   WARNING: Either you specified a nonexistent disk drive or you omitted the colon after a disk drive name. PRINT skips this file and processes any remaining files named on the command line. Check your spelling of the file names and try the command again.

9. `Invalid parameter`

   ERROR: You gave a switch that PRINT does not recognize, or you gave the /B, /D, /M, /Q, /S, or /U switch when you ran PRINT the second or subsequent time. If you gave an unknown switch, try the command again. If you attempted to use one of the mentioned switches, you must restart DOS and run PRINT again with the switch.

10. `filename is currently being printed`
    `filename is in queue`

    INFORMATION: This message tells you which file is being printed and names the files that are in line to be printed. The message appears when you use PRINT with no parameters or when you queue additional files.

11. `List output is not assigned to a device`

    ERROR: PRINT aborted because the command did not recognize the device you specified. To solve this problem, reissue PRINT and name a valid device when requested.

12. `filename Pathname too long`

    WARNING: The file specification for `filename` exceeded the 63-character limit. PRINT skips this file and processes any remaining files that are named on the command line. If the 63-character limit was exceeded, create

a new subdirectory closer to the root subdirectory, move the file to the new subdirectory (so that there are fewer subdirectories in the chain to the file), and try the PRINT command again. Otherwise, the command may have failed because you omitted a space. In that case, retype the command and try it again.

13. `PRINT queue is empty`

    INFORMATION: No files are in line to be printed by PRINT.

14. `PRINT queue is full`

    WARNING: You attempted to place too many files in the PRINT queue. The request to add more files fails for each file past the limit. You must wait until PRINT processes a file before you can add another file to the queue.

15. `Resident part of PRINT installed`

    INFORMATION: The first time you use PRINT, this message indicates that PRINT has installed itself in DOS and has increased the size of DOS by about 5,500 bytes.

16. `Disk-error-type error on file filename`

    WARNING: This message appears on the printout. A disk error (`disk-error-type`) occurred when DOS tried to read `filename`. PRINT follows its procedure for handling disk errors.

PROMPT *V2, V3, V4—Internal*

Customizes the DOS system prompt (`A>`, the A prompt).

Syntax

PROMPT *promptstring*

promptstring is the text to be used for the new system prompt.

Reference

See Chapter 6.

Rules

1. If you do not enter the *promptstring*, the standard system prompt reappears (`A>`).
2. Any text entered for *promptstring* becomes the new system prompt. You can enter special characters by using meta-strings.
3. The new system prompt stays in effect until you restart DOS or reissue the PROMPT command.
4. You can use the SET command to see the text of PROMPT after it has been set.

5. If you want to start the prompt with a DOS delimiter character (blank space, semicolon, or comma, for example), precede the character with a null meta-string. These null characters, such as $A, have no meaning to the PROMPT command.

Meta-Strings

A *meta-string* is a group of characters that has been transformed into another character or group of characters. To use certain characters (for example, the < or > I/O redirection symbols), you must enter the appropriate meta-string to place the desired characters in your *promptstring*. Otherwise, DOS immediately attempts to interpret the character.

All meta-strings begin with the dollar sign ($) and have two characters, including the $. The following list contains meta-string characters and their meanings:

Character	*Meaning*
$	$, the dollar sign
_(underscore)	newline, (moves to the first position of the next line)
b	|, the vertical *b*ar
e	The *E*scape character, CHR$(27)
d	The *d*ate, like the DATE command
h	The backspace character, CHR$(8), which erases the previous character
g	>, the *g*reater-than character
l	<, the *l*ess-than character
n	The curre*n*t disk drive
p	The current disk drive and *p*ath, including the current directory
q	=, the e*q*ual sign
t	The *t*ime, like the TIME command
v	The DOS *v*ersion number
Any other	Nothing or null; the character is ignored

Examples

1. **PROMPT**

   or

   PROMPT ng

   PROMPT sets the DOS system prompt to the normal prompt (A>). Here, the first example shows the command's default use. If you do not specify a

promptstring, the standard system prompt is restored. In the second example, **$n** is the letter of the current disk drive, and **$g** is the greater-than sign (>). If the current disk drive is A:, the new prompt becomes `A + A>`, or `A>`.

2. **PROMPT The current drive is $n:**

This command sets the new system prompt to

```
The current drive is A:
```

3. **PROMPT $p**

PROMPT sets the system prompt to the current disk drive and the path to the current directory. If you are in a subdirectory on your hard disk, for example, you might see a system prompt such as the following:

```
C:\DOS\UTILS
```

4. **PROMPT $A;$t;**

PROMPT sets the system prompt to the system time, such as

```
;12:04:12.46;
```

Note that a semicolon normally separates items on the DOS command line. To make the system prompt's first character a delimiter, a null meta-string must be used. Because **$A** does not have any special meaning for DOS, the system ignores this meta-string and makes the semicolon the first character of the new prompt.

5. **PROMPT $$**

This sets the system prompt to imitate the UNIX Bourne shell prompt:

```
$
```

Notes

The PROMPT command gives users greater control over their systems, and in this case, the system prompt. With hierarchical directories, you will find it helpful to be able to display the current path on your disk drive. For that reason, you may want to set the system prompt to

PROMPT $p

This command displays the current disk drive and the current path as the prompt.

You can use any text in the prompt, including text you may not normally think of, such as your name. If, for example, you are conscious of time and forgetful of dates, you could use the following form of the PROMPT command:

PROMPT $d $t $p $_Please type a command $

This command sets the system prompt to display the date, time, and current disk drive and path. The next line displays the message, `Please type a command`, as in the following:

```
Fri  8-11-1989  9:57:13.11 C:\DOS
Please type a command
```

You can use the last dollar sign to ensure that a space appears between the `d` in `command` and whatever you type at the prompt. Some text editors, like IBM Personal Editor, strip away spaces after the last printable character on a line. Because DOS ignores the dollar sign, however, you can use it as the last character and keep the space. New versions of DOS may force you to use a complete null meta-string, such as the **$A**.

For all its usefulness, the previous system prompt has two disadvantages. It is wordy and can distract other people who use your computer. Further, if the current disk drive is a floppy disk drive, you must always have a floppy disk in the drive.

Have fun experimenting with PROMPT; you can try it many times and not hurt anything. After you create a prompt you like, place it in your AUTOEXEC.BAT file, or create a separate batch file that can be executed after DOS has started. To cancel your new prompt and restore the old `A>` prompt, just reissue PROMPT without specifying a *promptstring*.

RECOVER

V2, V3, V4—External

Recovers a file with bad sectors or a file from a disk with a damaged directory.

Syntax

To recover a file, use the following form:

*dc:pathc***RECOVER** *d:path**filename.ext*

To recover a disk with a damaged directory, use this form:

*dc:pathc***RECOVER d:**

dc: is the disk drive holding the command.

pathc\ is the path to the command.

d: is the disk drive holding the damaged file or floppy disk.

path\ is the path to the directory holding the file that will be recovered.

filename.ext is the file that will be recovered.

Reference

See Chapter 5.

Rules

1. Do not use RECOVER on files or disks that do not need it.
2. If you do not give a drive name, DOS uses the current disk.
3. If you do not give a path name, DOS uses the current directory.
4. If you give a file name, DOS skips the bad sectors and recovers as much information as possible from that file.
5. If you give only a drive name, DOS attempts to recover the disk's directory. (This rule applies to the second syntax line shown earlier.)
6. If you do not specify a disk drive name, path name, or file name, RECOVER displays an error message and immediately exits.
7. If you give a wild-card character in the file name, RECOVER works only on the first file that matches the ambiguous name.
8. RECOVER does not restore erased files.
9. Do not use RECOVER on a networked disk drive. If you do, DOS responds with an error message.
10. Do not use RECOVER on a disk drive that has been tricked with an ASSIGN, SUBST, or JOIN command. The results are unpredictable.

Notes

RECOVER attempts to recover either a file with a bad sector or a disk with a directory containing a bad sector. To recover a file that has one or more bad sectors type **RECOVER d:filename.ext**. (DOS tells you if a file has bad sectors, by displaying a disk-error message when you try to use the file.)

When RECOVER works on a file, DOS attempts to read the file's sectors one at a time. After RECOVER successfully reads a sector, the information is placed in a temporary file. RECOVER skips any sectors that it cannot successfully read, but the FAT is marked so that no other program can use the bad sector. This process continues until the entire file has been read. RECOVER then erases the old file and gives its name to the temporary file, which becomes a new replacement file. This new file is placed in the directory where the old file resided.

If the damaged file is a program file, the program probably cannot be used. If the file is a data or text file, some information can be recovered. Because RECOVER reads the entire file, make sure that you use a text editor or word processor to eliminate any garbage at the end of the file.

To recover a disk, enter the disk drive name after the command. DOS creates a new root directory and recovers each file and subdirectory. The system names the recovered files FILEnnnn.REC ("nnnn" is a four-digit number). Even good files are placed in FILEnnnn.REC files. To determine which original file corresponds to a FILEnnnn.REC file, you must either use the TYPE command to view each file, or print each file and use the last printed directory of the disk, or have a good memory. A disk editor that can display a file's ASCII and hexadecimal characters is helpful. If you do not have a disk editor, use TYPE to view

each file and rename it after you have determined its identity. Then try to guess the names of any remaining files.

Remember that subdirectories also are recovered in FILEnnnn.REC files. When you TYPE a subdirectory file, you see that the first character is a period (.). A double period (..) then appears about halfway across the screen. When you see these two symbols, you can safely guess that this file is a subdirectory.

RECOVER does not recover erased files. You need a separate utility program to do this.

Messages

1. `Cannot RECOVER a Network drive`

   ERROR: You attempted to run RECOVER on a disk drive that is part of a network. You must pause or disconnect from the network and rerun RECOVER.

2. `File not found`

   ERROR: RECOVER cannot find the file you specified. Check the file name's spelling, make sure that the file exists on the specified disk drive and directory, and try the command again.

3. `Invalid drive or file name`

   ERROR: One of the following errors occurred: (1) you specified a nonexistent disk drive; (2) you omitted the colon between the disk drive and file names; (3) you misspelled the directory path; or (4) you failed to name a disk drive or file. Check each possibility and reissue the command.

4. `Invalid number of parameters`

   ERROR: You specified more than one argument for RECOVER. Perhaps you inserted a space between the disk drive name and path name or between the path name and file name, or you used an illegal character. Check the command line and try the command again.

5. `Press any key to begin recovery of the file(s) on drive d:`

   INFORMATION and WARNING: RECOVER is asking you to confirm that you want to recover a file or all files on disk drive `d:`. If you want RECOVER to attempt to recover the files on the disk drive, press any key. If you do not want to recover the files, press Ctrl-Break or Ctrl-C. Be sure to check the command line before pressing the key to begin. RECOVER displays this message for the recovery of a single file or all files.

6. `Warning – directory full`
   `nnn file(s) recovered`

   ERROR: The disk's root directory is full. No more files can be recovered, and RECOVER stops. You should either erase the files in the root directory

or copy them to another disk, to free more directory space. Then rerun RECOVER.

7. `xxxx of yyyy bytes recovered`

   INFORMATION: RECOVER is telling you the number of bytes it was able to salvage from a file.

RENAME or REN *V1, V2, V3, V4—Internal*

Changes the name of the disk file(s).

Syntax

RENAME *d:path***filename1**.*ext1* **filename2** *.ext2*

REN *d:path***filename1**.*ext1* **filename2**.*ext2*

d: is the disk drive holding the file(s) to be renamed.

path\ is the path to the file(s) to be renamed.

filename1.*ext1* is the file's current name. Wild cards are allowed.

filename2.*ext2* is the file's new name. Wild cards are allowed.

Reference

See Chapter 7.

Rules

1. If you do not specify a drive name, DOS uses the current disk drive.
2. If you do not specify a path name, DOS uses the current directory.
3. You can give a disk name and path name for the first file name only.
4. You must give both the previous and the new file names and any appropriate extensions.
5. Wild-card characters are permitted in the file names.

Notes

RENAME, or the short form REN, changes the name of a file on the disk. The command's form is

RENAME old_name new_name

Because you are renaming an established disk file, the file's drive or path designation goes with the old name so that DOS will know which file to rename.

Wild-card characters are acceptable in either the old or the new name. Be careful, however, when you use these characters, because they can be troublesome. For example, suppose that a directory has the following files:

LETTER.TXT
LETTER1.TXT
MEMO.DOC
MEMO.TXT

If you attempt to rename all .TXT files as .DOC files with the command

RENAME *.TXT *.DOC

RENAME stops when it encounters MEMO.TXT, because a MEMO.DOC already exists. All .TXT file names listed before MEMO.TXT, however, are changed to .DOC file names.

Messages

1. `Duplicate filename or File not found`

   ERROR: You attempted to change a file name to a name that already exists, or you asked DOS to rename a file that does not exist in the directory. Check the directory for conflicting names. Make sure that the file name exists and that you have spelled it correctly. Reissue the command.

2. `Invalid number of parameters`

   ERROR: You gave too few or too many file names. You must give two file names: the file's existing name and the new name that the file will receive. Make sure that you did not insert spaces between the disk drive name and path name or between the path name and file name for the first file. If this is not the case, you may have used an invalid character in one or both file names.

3. `Missing file name`

   ERROR: You forgot to enter the new name for the file.

REPLACE *V3.2, V3.3, V4—External*

Replaces files on one disk with files of the same name from another; adds files to a disk by copying them from another disk.

Syntax

*dc:pathc***REPLACE** *ds:paths***filenames**.*exts dd:pathd /A/P/R/S/W*

With DOS V4, you can add a */U* switch:

*dc:pathc***REPLACE** *ds:paths***filenames**.*exts dd:pathd*
/A/P/R/S/W/U

dc: is the disk drive holding the command.

pathc\ is the path to the command.

ds: is the disk drive holding the replacement file(s).

paths\ is the path to the replacement file(s).

filenames.*exts* is the name of the replacement file(s). Wild cards are allowed.

dd: is the disk drive whose file(s) will be replaced.

pathd is the path to the directory that will receive the replacement file(s).

In the fragment *ds:paths***filenames**.*exts*, *s* represents the *source*. The source is the file that will be added to a directory, or that will replace an existing file in the directory.

In the fragment *dd:pathd*, *d* represents the *destination*. The destination can be either the file that will be replaced, or the disk and directory where a new file will be added. DOS refers to the destination as the *target*.

Switches

/A *Adds* files from the source disk that do not exist on the destination disk.

/P Displays a *prompt* asking if the file should be replaced or added to the destination.

/R Replaces files on the destination disk even though the files' *read-only* attribute is on.

/S Replaces all files with matching names in the current directory and its subordinate *subdirectories*. The /S switch does not work with the /A switch.

/W Causes REPLACE to prompt and *wait* for the source floppy disk to be inserted.

/U Replaces only those files whose date and time are earlier than the source files' date and time (DOS V4 and later).

Exit Codes

REPLACE returns the DOS exit codes. A zero exit code indicates successful completion, and nonzero exit codes indicate various types of errors. Common exit levels are these:

0 Successful operation.

2 No source files were found.

3 Source or target path is invalid.

5 Access denied to the file(s) or directory(ies).

8 Out of memory.

11 Invalid parameter or incorrect number of parameters.

15 Invalid disk drive.

22 Incorrect version of DOS.

Reference

See Chapter 5.

Rules

1. If you do not name the source disk drive, DOS uses the current disk drive.
2. If you do not name the source path, DOS uses the current directory.
3. You must specify a source file name. Wild cards are allowed.
4. If you do not name the destination disk drive, DOS adds files to—or replaces files on—the current drive.
5. If you do not name the destination path, DOS adds files to—or replaces files in—the current directory.
6. REPLACE adds or replaces files according to their file name, not their contents. This means that REPLACE can replace program or data files whose names match, even if their contents are different.
7. Use the /A switch to add files to the destination directory. When you use the /A switch, REPLACE compares the source and destination directories. If REPLACE finds any files in the source directory that are not in the destination directory, it copies those files into the destination directory. This form of REPLACE, however, cannot work with more than one destination subdirectory at a time. Therefore, the /S and /A switches cannot be used together.
8. Use the /S switch to replace files in more than one directory on the destination. REPLACE searches the specified directory and all subordinate subdirectories, and replaces all files whose names match files in the source directory.
9. The /A and /S switches cannot be used together. All other switches, however, can be used together.
10. Use the /P switch to add or replace files selectively. REPLACE displays a prompt asking if each file should be added or replaced. You can respond to the prompt by pressing Y for yes or N for no.
11. Use the /R switch to replace read-only files. If this switch is not used, REPLACE skips files marked with the read-only flag.
12. If you use the /W switch, REPLACE prompts you to insert the source floppy disk. This switch is helpful if your computer has only one floppy disk drive, or if you must change floppy disks before REPLACE searches for the source file names. When you give the /W switch, REPLACE waits for you to press a key before it begins reading the source files. You may notice that when you use the /W switch, REPLACE responds with a slightly different message than when you use the /A switch. If you add files (/A), REPLACE prompts you to press any key to begin adding files. When you use the /W switch without the /A switch, REPLACE displays a message to press any key to begin replacing files. If the /W switch is not used, REPLACE assumes that

the correct floppy disk already is in the drive and immediately starts searching for the source files.

13. If you have DOS V4.0 or V4.01, you can use the /U switch to replace files whose date and time are older than the source files' date and time. This switch is useful only if you set the system date and time whenever you start DOS, or if your computer has an automatic battery-powered clock that sets the date and time for you.
14. REPLACE does not process hidden or system files.

Examples

1. **REPLACE A:*.COM C:\BIN**

   REPLACE searches the current directory of drive A: for files with a .COM extension, then copies them over those files with matching names in C:\BIN. Any .COM file on A: is not copied if its name does not match a file in the directory C:\BIN.

2. **REPLACE A:*.COM C:\BIN /S**

   REPLACE searches the current directory of drive A: for files with a .COM extension, then copies them over those files with matching names in C:\BIN. Because the /S switch has also been given, however, REPLACE also replaces all matching files in the lower subdirectories of C:\BIN.

3. **REPLACE A:*.COM C:\BIN /S/R**

   This example is identical to the second example, but because the /R switch has been added, REPLACE now also replaces read-only files. Remember, however, that REPLACE does not process hidden or system files. Imagine trying to process a start-up diskette, for example, which contains the files IO.SYS and MSDOS.SYS. REPLACE does not process these two files. Although the two files are marked read-only, they also are marked as hidden and system files.

4. **REPLACE A:*.COM C:\BIN /P**

   This command replaces the files on C:\BIN that match files on the current directory of drive A:, but REPLACE first prompts you to confirm each file individually. The /P switch is used when you need more control in determining which files should be replaced.

5. **REPLACE A:*.COM C:\BIN /A**

   This command uses the /A switch to add new files to the destination subdirectory. REPLACE searches the current directory of drive A: for files that have a .COM extension, but that do not match any of the existing file names in C:\BIN. These non-matching files are copied from A: to C:\BIN.

6. **REPLACE A:*.* C:\ /S/W/P**

   This command replaces files on C: with files of the same name in the current directory of A:. REPLACE prompts you to insert the floppy disk in A: before

REPLACE reads the source files (the effect of the /W switch). The /P switch forces REPLACE to confirm that individual files should be replaced. You should be aware, however, that this command can replace a different file in some directory of the disk, if that disk file has the same name as a file on the source diskette.

7. **REPLACE A:*.* C:\DOC /S/U**

   This command searches C:\DOC and any subordinate subdirectories and replaces all files whose date and time are older than the date and time of the files in drive A:. This is a good way to update your files if you set the date and time whenever you start DOS.

Notes

The REPLACE command, which was added in DOS V3.2, is a semi-intelligent COPY program that can selectively update or add files to one or more directories.

You can use REPLACE to replace one or a few files that are duplicated in several subdirectories. You also can use the command to update a major application program, or even DOS itself. REPLACE is especially useful because it can search subdirectories for files to replace.

REPLACE replaces or adds files based on the matching of file names. The command disregards drive names, directory names, and file contents, searching only for the appropriate file names and extensions. By using this exclusive search facility, you can quickly find older files and replace them with their newer counterparts.

If you do not use the REPLACE command with caution, however, its speedy "find and replace" capability can have the effect of an unrelenting "search and destroy" mission on your data. Be careful when you unleash REPLACE on several subdirectories at a time, particularly when you use REPLACE /S on the entire disk. You could wind up replacing a file that you want to save somewhere on the disk, because REPLACE updates the file based on file name alone.

To prevent such unwanted replacements, you should limit the destination path name to cover only the directories holding the files you want REPLACE to replace. Check the source and destination directories for matching file names. If you find any conflicts or have doubts, use the /P switch; REPLACE will ask for your approval before replacing any files.

If you have DOS V4.0 or V4.01, the /U switch may help you avoid replacing wrong files. The /U switch compares the files' date and time stamp. The destination file is replaced only if it is older than the source file.

Messages

1. `Access denied filename`

   ERROR: REPLACE attempted to replace a file (`filename`) that is marked as read-only, but the /R switch was not used. Otherwise, the file is being used by another program or computer and is marked temporarily as read-

only. In either case, the problem is with the destination file. If `filename` is the source file, another computer or process has temporarily marked the source file as write-only.

If you want to replace a file whose read-only attribute is set, you can reissue the REPLACE command with the /R switch. You also can use the ATTRIB command to turn off the read-only flag of the file that will be replaced (the destination file, not the source file). If this file is being used by another program or computer, wait until the other program or computer is finished and run the REPLACE command again.

2. `Add filename? (Y/N)`
   `Adding filename`

   INFORMATION and WARNING: These messages appear if the /A and /P switches are used. In the first message, REPLACE asks whether the file `filename` should be added to the destination directory. If you press Y for yes, REPLACE proceeds and the second message informs you that REPLACE is adding `filename`. If you answer N for no, REPLACE does not add the file and the second message does not appear.

3. `File cannot be copied onto itself filename`

   WARNING: The source and destination disk and directories are identical. You probably forgot to specify a destination, so the source disk and directory are the current disk and directory. Otherwise, you specified the same disk drive and directory twice. REPLACE does not process `filename`.

   Check the command line to make sure that you specified the correct source and destination for REPLACE, then try the command again.

4. `nnn file(s) added`

   or

   `nnn file(s) replaced`

   INFORMATION: REPLACE indicates how many files were successfully added or replaced. The first message appears when the /A switch is used; the second message appears when the /A switch is not used. The message does not indicate that all potential files were added or replaced successfully. Rather, the message appears when at least one file has been added or replaced successfully, regardless of any errors that occurred later.

5. `No files added`

   or

   `No files replaced`

   ERROR: REPLACE did not add or replace any files. The first message appears when you give the /A switch; the second message appears when you do not give the /A switch. One of the following errors probably occurred: (1) you misspelled the source file name or the destination name; (2) you gave the wrong source name; (3) you gave no destination name or

the wrong one; or (4) you put the wrong floppy disk in the disk drive. Otherwise, REPLACE might not have been capable of adding or replacing files because of insufficient disk space.

Make sure that the command line is properly phrased, that the correct floppy disk is in the disk drive, and that the destination disk has sufficient room for the files. Try the command again.

6. `Invalid parameter`

   ERROR: You either gave a switch that REPLACE does not recognize or forgot to type the switch character (/). Check the command line and try the command again.

7. `Invalid drive specification d:`

   ERROR: You gave a nonexistent disk drive name (`d:`) for either the source or the destination. Check the command line and try again.

8. `Insufficient disk space - filename:`

   WARNING: The file `filename` could not be added or replaced. If you gave the /A switch, the destination disk did not have room for the file. If you did not give the /A switch, the replacement file was larger than both the old file and the remaining free space on the destination disk. REPLACE skips this file and continues.

   To solve this problem, you can erase any unwanted files from the destination disk and use the COPY command on the files that REPLACE did not copy. To add more new files, erase any unwanted files from the destination disk and give the REPLACE /A command again.

9. `No files found filename`

   ERROR: REPLACE could not find any files that matched the source file name `filename`. One of the following errors probably occurred: (1) you misspelled the source file name; (2) you gave the disk drive and directory name but omitted the file name; (3) you gave the wrong disk drive or directory name for the source; or (4) you put the wrong floppy disk in the disk drive. Check the command line and make sure the correct floppy disk is in the disk drive, then try the command again.

10. `Parameters not compatible`

    ERROR: You gave both the /A and /S switches, which cannot be used together in the same REPLACE command. If you want to replace files, omit the /A switch. Remember that you cannot add files to more than one directory at a time, so you cannot give the /S switch along with the /A switch. If you want to add files to more than one directory, you must issue separate REPLACE commands, each time specifying a different directory to which files should be added.

11. `Path not Found pathname`

    ERROR: REPLACE could not find the destination directory path, `pathname`. Either the destination directory name is misspelled or you gave the wrong disk drive name. Try the command again and give the correct destination disk drive name and directory name.

12. `Press any key to begin adding file(s)`

    `Press any key to begin replacing file(s)`

    INFORMATION: You used the /W switch, and REPLACE is waiting until you strike a key before reading the source files. The first message (adding files) appears if the /A switch is used; the second message (replacing files) appears if the /A switch is not used. Press any key to make REPLACE continue.

13. `Replace filename? (Y/N)`
    `Replacing filename`

    INFORMATION and WARNING: These messages appear when the /P switch is used without the /A switch. The first message asks whether the file should be replaced on the destination directory. If you type Y for yes, REPLACE replaces the old file with the file from the source, then displays the second message. If you answer N for no, REPLACE does not add the file, and the second message does not appear.

14. `Source path required`

    ERROR: You either mistyped the command or omitted the source file name. You must specify a source file name to REPLACE. Check the command line and try again.

RESTORE

V2, V3, V4—External

Restores one or more backup files from one disk onto another disk.

Syntax

dc:pathc\ **RESTORE d1:** *d2:path/filename.ext /S /P /M /N /B:date /A:date /L:time /E:time*

dc: is the disk drive holding the command.

pathc\ is the path to the command.

d1: is the disk drive holding the backup files.

d2: is the disk drive that will receive the restored files.

path\ is the path to the directory that will receive the restored files.

filename.ext is the file that you want to restore. Wild cards are allowed.

Switches

/S	Restores files in the current directory and all its subordinate *subdirectories*. When this switch is given, RESTORE re-creates any necessary subdirectories that have been removed, then restores the files in the re-created sub-directories.
/P	Causes RESTORE to *prompt* for your approval before restoring a file that has been changed since the last backup, or before restoring a file that is marked as read-only.
/M	Restores all files that have been *modified* or deleted since the backup set was made.
/N	Restores all files that *no longer* exist on the destination.
/B:date	Restores all files that were created or modified on or *before* the *date* you specify.
/A:date	Restores all files that were created or modified on or *after* the *date* you specify.
/L:time	Restores all files that were created or modified at or *later* than the *time* you specify.
/E:time	Restores all files that were created or modified at or *earlier* than the *time* you specify.

Exit Codes

0	Normal completion.
1	No files were found to restore.
3	Terminated by the operator (through a Ctrl-Break or Esc).
4	Terminated by an encountered error.

Reference

See Chapter 7.

Rules

1. You can use the RESTORE command only on files that you have saved with the BACKUP command.
2. You must give the name of the disk drive holding the backup files. If the current disk is the disk to receive the restored files, you do not need to specify the destination disk drive.
3. If you do not name a path, RESTORE uses the current directory of the receiving disk.
4. If you do not give a file name, RESTORE restores all backup files from the directory. Not giving a file name is the same as using *.*.

5. RESTORE prompts you to insert the backup diskettes in order. If you insert a diskette out of order, RESTORE prompts you to insert the correct diskette.

6. Use the /B:date switch to restore files that were modified or created on or before a given date. Similarly, you can use the /A:date switch to restore files that were created or modified on or after a given date. When you use these switches, enter the date in the same format used by the DOS DATE command.

7. For greater control in determining which files to restore, use the /E:time switch. This switch enables you to restore files that were modified or created before the time you specify. Similarly, you can use the /L:time switch to restore files that were modified or created after the given time. When you use these switches, enter the time in the same format used by the DOS TIME command.

8. You should not use the /E:time or /L:time switch without also using the /B:date or /A:date switch. Otherwise, files that match the specified time will be restored regardless of their date. You could actually restore an older file over a more recent file.

9. The /M switch restores files that have been modified, do not exist, or have been deleted since the backup set was made. A file has been modified if its archive flag is on.

10. The /N switch causes RESTORE to compare the backup and destination disks, then copy to the destination disk any files that the backup set has but the destination disk does not have.

11. Do not combine the /B, /A, and /N switches in the same RESTORE command.

12. Be cautious when you restore files that were backed up while an ASSIGN, SUBST, or JOIN command was in effect. When you use RESTORE, clear any existing APPEND, ASSIGN, SUBST, or JOIN commands. Do not use RESTORE /M or RESTORE /N while APPEND /X is in effect. RESTORE attempts to search the directories for modified or missing files. APPEND tricks RESTORE into finding files in the paths specified to the APPEND command. RESTORE then may restore files that it should not, and not restore files that it should. Give the **APPEND;** command to disable APPEND.

Notes

BACKUP and RESTORE V3.3, V4.0, and V4.01 are radically different from earlier versions. BACKUP V3.3 and later versions place all backed-up files in one larger file and maintain a separate information file on the same disk. In V3.3 and later, RESTORE handles the new and old backup file formats, which means that these later versions of RESTORE will restore backups created by any version of BACKUP.

The /M switch restores files that have been modified since the last backup (the archive flag is on), and files that have not existed in the same directories since the backup set was made. The /M switch is best used in a disastrous circumstance, for example, when the current copy of your modified files is bad and you have erased files from your hard disk.

The /N switch restores files that are not in the same directory as they were when the backup set was made. If a file has been erased or moved (copied and then erased), the /M or /N switch restores the file. If you are attempting to restore erased files, use the /N switch, then the /M switch.

The /P switch causes RESTORE to prompt for your approval before restoring a modified or read-only file. Use the /P switch to restore files selectively, especially when you use /M, which can restore an out-of-date backup file over a modified file.

Unlike earlier versions, RESTORE V3.3 and subsequent versions do not restore the DOS system files IO.SYS, MSDOS.SYS, and COMMAND.COM. If you reformat a disk with the /S switch, FORMAT automatically places these three files on the disk. You also can restore these files with the SYS command, transferring IO.SYS and MSDOS.SYS to the disk, then copying COMMAND.COM from your DOS start-up disk.

Messages

1. `*** Files were backed up mm/dd/yy ***`

   INFORMATION: RESTORE displays the date on which the files were backed up. The date's format varies depending on your country code's setting.

2. `filename File creation error`

   ERROR: DOS found an error while attempting to restore the file `filename`, and RESTORE stops. You should run CHKDSK on the receiving disk to see if it has physical problems. If any messages appear, follow the appropriate directions under CHKDSK in the Command Reference.

3. `Insert backup diskette nn in drive d:`
   `Strike any key when ready`

   INFORMATION: RESTORE wants the next floppy disk in sequence. This message appears when you are restoring files that were backed up onto floppy disks. Insert the next floppy disk (in the proper sequence) in drive `d:` and press any key.

4. `Insert restore target in drive d:`
   `Strike any key when ready`

   RESTORE is asking you to insert the floppy disk that will receive the restored files. This message appears only when you are restoring files onto floppy disks. Insert the target floppy disk in drive `d:` and press any key.

5. `Invalid drive specification`

   ERROR: You failed to specify the disk drive holding the backup files, or you named a nonexistent disk drive. Otherwise, a resident (background) program may be interfering with RESTORE.

6. `Invalid number of parameters`

   ERROR: You failed to specify the disk drive holding the backup files, or you gave too many drive or file names. You also may have inserted a space or other illegal character in a file name.

7. `Invalid parameter`

   ERROR: You gave a switch that RESTORE does not recognize, or you gave an invalid path or file name for the disk drive holding the backup files (**d1:**). Check the command line and try again.

8. `Invalid path`

   ERROR: RESTORE was unable to create or use a subdirectory in another subdirectory. Run CHKDSK and follow the directions in the Command Reference.

9. `*** Not able to restore file ***`

   WARNING: RESTORE could not restore the file because another computer or program is using the file. If this is not the case, you probably stopped RESTORE with a Ctrl-Break sequence. If you did not stop RESTORE, you should have the other computer or program stop after RESTORE is finished, then rerun RESTORE just for this file.

10. `*** Restoring files from drive d: ***`

    INFORMATION: RESTORE is getting the backup files from the disk or floppy disk in drive `d:`.

11. `Source does not contain backup files`

    ERROR: RESTORE could not find any files that were backed up with the BACKUP command. BACKUP may have malfunctioned when it was backing up files, or you inserted the wrong floppy disk if you were restoring from floppy disks. Otherwise, BACKUP could not find the \BACKUP subdirectory, if you were restoring files from a hard disk.

    Run DIR to get a directory listing of the floppy disk and the hard disk. Look for the file BACKUPID.@@@ on the floppy disk. If this file is missing, either the floppy disk is not a backup or BACKUP malfunctioned (DOS V3.0 only). If the floppy disk is a backup, you cannot restore the files. Use your previous backup set.

    Look for the \BACKUP subdirectory in the hard disk's root directory. If the subdirectory is missing, the backup files also are missing. Restart DOS and check again. Run CHKDSK if necessary. If the directory is still missing, the backup files have been lost.

12. `Source and target drives are the same`

    ERROR: RESTORE determined that the disk drive that is holding the backup files is the same as the drive that you designated to receive the restored files. You may have forgotten to specify the disk drive holding the backup files or the target disk. If you have a system with one floppy disk drive and you are attempting to restore files onto a floppy disk, specify drives A: and B:.

13. `System files restored`
    `Target disk may not be bootable`

    WARNING: You have restored the three system files (IO.SYS, MSDOS.SYS, and COMMAND.COM) from the backup floppy disks. If these files are from a previous version of DOS, the receiving disk cannot be used to start (boot) DOS. If you know that the system files were from the version you are currently using, ignore this message. Otherwise, run SYS on the disk to put current versions of these three files on the disk.

14. `Target is full`
    `The last file was not restored`

    ERROR: RESTORE stopped before the current file (listed in the line above this message) was fully restored, because the receiving disk is full. RESTORE deletes the partially restored file.

15. `Warning! Diskette is out of sequence`
    `Replace the diskette or continue if okay`
    `Strike any key when ready`

    WARNING: You inserted a backup floppy disk out of order. Place the correct floppy disk in the drive and continue. You may continue with the wrong floppy disk if you do not want to restore any files and if you are not in the middle of a large file (backed up onto two or more floppy disks) that has been partially restored. If a file has been partially restored, RESTORE will want to restore the rest of the file. RESTORE continues displaying this message until the right floppy disk is inserted.

16. `Warning! File filename`
    `was changed after it was backed up`
    `or is a read-only file`
    `Replace the file (Y/N)?`

    WARNING: This message appears when you give the /P switch. The file `filename` already exists on the hard disk and is marked as read-only. Otherwise, the file's date is later than that of the backup copy, which means that the backup copy may be out of date. Answer Y to replace the preexistent file with the backup copy or N to skip that file.

17. `Warning! No files were found to restore`

    WARNING: RESTORE searched the backup floppy disks but did not find any files matching the file names you specified. This message also can occur if you reset SWITCHAR.

18. `Unrecoverable file sharing error`

    WARNING: With SHARE.EXE loaded, RESTORE cannot restore a file because another program is using the file or because the file-sharing table is full. The file is not restored.

RMDIR or RD

V2, V3, V4—Internal

Removes a subdirectory.

Syntax

RMDIR *d:***path**

RD *d:***path**

d: is the drive holding the subdirectory.

path is the path to the subdirectory. The last path name is the subdirectory you want to delete.

Reference

See Chapter 6.

Rules

1. If you do not specify a drive name, DOS uses the current drive.
2. You must name the subdirectory to be deleted. If you supply a path, its last name must identify the subdirectory to be deleted.
3. The subdirectory to be deleted must be empty (excluding the . and .. entries).
4. You cannot delete a disk's current directory or root directory.

Notes

RMDIR, or the short form RD, removes subdirectories from the disk. RMDIR is the opposite of MKDIR (make directory).

When you remove a subdirectory, it must be empty except for the current directory file (.) and any parent directory files (..). You cannot remove the current directory. To do this, you must first move to another directory, then use RMDIR to remove the previously current directory. If you attempt to remove a subdirectory that is not empty or that is the current directory, DOS displays an error message and does not delete the directory.

Message

```
Invalid path, not directory
or directory not empty
```

RMDIR did not remove the specified directory because one of the following errors occurred: (1) you gave an invalid directory in the path; (2) the subdirec-

tory still has files in it other than the . and .. entries; or (3) you misspelled the path or directory name to be removed. Check each possibility and try again.

SELECT *V3, V4—External*

Prepares a disk with the DOS files and configures the CONFIG.SYS and AUTOEXEC.BAT files for your country. For DOS V4, SELECT was expanded to a full-featured, menu-oriented DOS installation utility.

Syntax

For DOS V3.0 through V3.3, use the following form:

*dc:pathc***SELECT** *ds: dd:pathd* **countrycode keycode**

dc: is the disk drive holding the command.

pathc\ is the path to the command.

ds: is the source disk.

dd:pathd\ is the destination disk and subdirectory.

Reference

See Chapter 15.

Rules

1. SELECT uses the FORMAT command to format the target disk (the disk that will receive the DOS files). Do not use SELECT on a disk that contains usable information.
2. You should use your original DOS start-up floppy disk, master floppy disk, or a copy of it. It is best to use a copy of the start-up floppy disk with a tab over the write-protect notch.
3. If you are using a floppy disk as the destination disk, make sure that the floppy disk you use is appropriate to your system's floppy disk drive. That is, use only a 1.2M floppy disk in a 1.2M disk drive, a 1.44M floppy disk in a 1.44M disk drive, and so on.
4. If you do not specify a source disk drive, DOS uses drive A: as the source drive.
5. If you do not specify a source disk drive or a destination disk drive, DOS uses drive B: as destination drive.
6. If you specify a source disk drive, you must also specify a destination disk drive.

7. If you specify only one disk drive, DOS uses A: as the source. The named disk drive is used as the destination.
8. A: and B: are the only valid source disk drives. SELECT responds with an error message if you use any other disk drive name as the source.
9. SELECT accepts any valid disk drive as the destination drive, but the source and destination drives must be different.
10. If you do not specify a destination directory, SELECT copies the DOS files to the destination disk's root directory.
11. If you specify a destination directory, the path (*pathd*) must be an absolute path; that is, it cannot contain the . or .. entries. SELECT formats the disk and creates the directories you specify.
12. You do not need to use the SELECT command if your start-up disk already contains the CONFIG.SYS and AUTOEXEC.BAT files. Simply edit the CONFIG.SYS and AUTOEXEC.BAT files to include the proper entries, and copy the COUNTRY.SYS and KEYBOARD.SYS files to the disk, preferably to the root directory of the disk.

Rule for V4.0 and V4.01:

13. SELECT is used when you start your computer with the DOS Install disk. Select is menu-driven and configures DOS for the computer according to your specifications.

Notes

The SELECT program helps you set up DOS on a new disk. SELECT V3.3, which is different from earlier versions of SELECT, works with any hard or floppy disk. SELECT V3.2 and earlier versions work only with floppy disks.

SELECT V3.3 displays a warning that the program will format your disk, then prompts you for confirmation. SELECT then runs the FORMAT /S command to format the disk and place the DOS system files on the disk. If a hard disk is the destination, SELECT runs the FORMAT /S /V command to place a volume label on the hard disk. DOS presents the normal FORMAT messages, including the message to insert the floppy disk in the correct disk drive, or the warning in response to your request to format the hard disk. If you are not sure whether you should run SELECT, you can type the Ctrl-Break or Ctrl-C sequence to stop FORMAT and SELECT.

After FORMAT is complete, SELECT executes XCOPY to copy the remaining DOS files to the disk's root directory, or, if directed, to an optional subdirectory. This process includes the COUNTRY.SYS, KEYBOARD.SYS, and KEYB.COM files, which are used for international language support.

SELECT then establishes a CONFIG.SYS file in the root directory with the line

```
COUNTRY=countrycode
```

`countrycode` is the country code specified to the command.

Next, SELECT creates an AUTOEXEC.BAT file in the disk's root directory with the following lines:

```
PATH \;\pathd
KEYB keycode codepage
ECHO OFF
DATE
TIME
VER
```

This batch file establishes the DOS PATH command, issues the appropriate KEYB command for international language use of the keyboard, turns off ECHO, and issues the DATE, TIME, and VER commands. The last three commands direct DOS to go through the same start-up procedure as if an AUTOEXEC.BAT file did not exist; the commands ask for the date and time, then display the DOS sign-on message.

In the PATH command line, *pathd* is the optional destination directory given to the SELECT command. If *pathd* is not given, SELECT uses the destination disk's root directory as the path. If a hard disk is the destination disk, the disk drive name is inserted before each directory for the PATH. For example, if C: is the destination disk and the destination directory is BIN, the PATH command used in the AUTOEXEC.BAT file will be

```
PATH C: \;C:\BIN
```

If no destination directory is given, the PATH is C:\. SELECT omits the second path specification in the PATH command.

SELECT creates a KEYB command in the AUTOEXEC.BAT file and gives the `keycode` to KEYB. SELECT also translates the `keycode` and specifies the appropriate `codepage` (font file) for the keyboard code. Code pages are explained in the KEYB command section of the Command Reference.

Although SELECT does help in setting up a new disk, this command wipes out any useful information previously on the disk. Therefore, do *not* use SELECT on an established disk!

If you prepare a floppy disk with SELECT, do not specify a path name for the destination if you use 360K or 720K floppy disks. The additional space taken by the subdirectory would be better used for storing files. Further, you might find it inconvenient to run DOS from a subdirectory on a floppy-based system.

For V4.0 and V4.01, SELECT has been modified even further. In these versions, SELECT is the program that starts when you boot your computer from the DOS Install disk. SELECT V4 enables you to configure and format your hard disk, install the DOS Shell and set up the CONFIG.SYS and AUTOEXEC.BAT files.

Messages

1. `Cannot execute A:program_name`

   ERROR: SELECT could not find `program_name` on drive A:. The file `program_name` may be FORMAT or XCOPY for DOS V3.2 and later, DISKCOPY for DOS V3.1 and earlier, or DISKCOMP.COM for DOS V3.0. You probably started SELECT without the DOS start-up or master floppy disk in drive A:. Check the floppy disk in drive A: and try again.

2. `File creation error`

   ERROR: SELECT could not create the CONFIG.SYS or the AUTOEXEC.BAT file. This error usually occurs because the target floppy disk (the floppy disk prepared by SELECT) has a flaw. Run SELECT again. If the the message reappears, use a different blank floppy disk as the target.

3. `Failure to access COUNTRY.SYS`
   `Failure to access KEYBOARD.SYS`

   ERROR: SELECT could not open the COUNTRY.SYS file (first message) or the KEYBOARD.SYS file (second message) on the source floppy disk to verify the country or keyboard code. Either the wrong floppy disk is in the disk drive, or you specified the wrong source disk drive. If you gave the wrong source disk drive, try the command again.

   If you believe that the correct floppy disk is in the disk drive, issue a DIR command and then look for the COUNTRY.SYS and KEYBOARD.SYS files. If these files do not appear, remove the floppy disk and insert a copy of your DOS floppy disk that contains these files, then reissue the command.

4. `Incorrect number of parameters`

   ERROR: You omitted the country code or the keyboard code. Otherwise, you forgot to leave a space between the disk drive names, the country code, and the keyboard code. Make sure that the command line is properly phrased and that you have given both the country and keyboard codes, then try the command again.

5. `Invalid country code`
   `Invalid keyboard code`

   ERROR: You omitted the country code or keyboard code. Otherwise, you forgot to leave a space between the disk drive names, the country code, and the keyboard code. Make sure that the command line is properly phrased and that you have given both the country and keyboard codes, then try the command again.

6. `Invalid drive specification`

   ERROR: One of the following errors occurred: (1) you named a nonexistent disk drive for the source or destination; (2) you used a disk drive name other than A: or B: for the source disk drive; (3) you omitted the

source disk drive and named A: as the destination disk drive; or (4) you specified the same disk drive for both the source and destination.

7. `Invalid parameter`

   ERROR: You forgot to place a colon after the source or destination disk drive name. Check the command line and reissue the command.

8. `Invalid Path`

   ERROR: The destination directory path is misspelled, has incorrect characters, or has more than 63 characters. Check the destination path name and try the command again.

9. `Invalid signature in COUNTRY.SYS file`
   `Invalid signature in KEYBOARD.SYS file`

   ERROR: While verifying the country or keyboard information in the COUNTRY.SYS or KEYBOARD.SYS file, SELECT determined that the named file was incorrect. The COUNTRY.SYS or KEYBOARD.SYS file exists but is damaged. The floppy disk is bad, the file has been altered, or another file has been copied over the COUNTRY.SYS or KEYBOARD.SYS file.

   Make another copy of your original DOS start-up floppy disk and try to copy the correct .SYS file over the invalid file. If no error message appears when you copy the file, reissue the SELECT command. If another error message appears, reformat or retire the old copy of the original DOS start-up floppy disk. If the `invalid signature` message appears again, you probably have a bad copy of DOS. See your dealer for a replacement.

10. `Read error, COUNTRY.SYS`
    `Read error, KEYBOARD.SYS`

    ERROR: While reading the COUNTRY.SYS or KEYBOARD.SYS file to verify country or keyboard information, SELECT encountered a disk error. Either the floppy disk or the floppy disk drive is bad.

    Make another copy of your original DOS start-up floppy disk and try to copy the correct .SYS file over the invalid file. If no error message appears when you copy the file, reissue the SELECT command. If another error message appears, reformat or retire the old copy of your DOS start-up floppy disk and make another copy of the original start-up floppy disk. If you can make a copy of the original start-up floppy disk, the disk drive works correctly and you should try the SELECT command again with this new copy. If DISKCOPY reports an error with the source floppy disk, the disk drive probably is bad.

11. `SELECT is used to install DOS the first time. SELECT erases everything on the specified target and then installs DOS. Do you want to continue (Y/N)?`

    INFORMATION and WARNING: This message appears when you start SELECT and warns you that SELECT will format the disk and copy the needed files to the disk. Be sure that the DOS start-up floppy disk is in the

correct disk drive. Answer Y if you want to continue or N if you do not want to continue.

12. `Unable to create directory`

    ERROR: SELECT formatted the destination disk but was unable to create the destination directory. The destination directory path probably contains invalid characters or is a reserved device name. Check the spelling of the destination path name and try the command again.

SET *V2, V3, V4—Internal*

Sets or shows the system environment.

Syntax

To display the environment, use the following form:

SET

To add to or alter the environment, use this form:

SET name=*string*

name is the string you want to add to the environment.

string is the information you want to store in the environment.

The *environment* is the portion of RAM reserved for alphanumeric information that can be examined and used by DOS commands or user programs. For example, the environment usually contains COMSPEC, which is the location of COMMAND.COM; PATH, the additional paths for finding programs and batch files; and PROMPT, the string defining the DOS system prompt.

Reference

See Chapter 8.

Rules

1. If you do not specify a name or a string, SET displays the current environment.
2. To place a string in the environment or to change the string associated with a current name, use **SET name**=*string*.
3. To delete a name, use **SET name=** without specifying a *string*.
4. Any lowercase letters in **name** are changed to uppercase letters when placed in the environment. The characters in *string* are not changed.
5. You can use the SET command rather than PROMPT or PATH to set the system prompt and the information for the PATH command.
6. You can use SET to change COMSPEC, or the location of COMMAND.COM.

7. When a resident program (such as MODE, GRAPHICS, or PRINT) is loaded, DOS cannot expand the environment past 127 characters. If you have not used 127 characters, you can add information up to that limit.

Notes

SET puts information in memory for later use by invoked batch files or programs. The command is most commonly used to store the directory path to data files or program overlays. An invoked program examines the RAM where the SET information is stored, then issues the proper commands to find the needed data or program overlays.

SET is a versatile command that can put almost any environment information into RAM. Unfortunately, few programs take advantage of this command's features.

SHARE *V3, V4—External*

Enables DOS support for file and record locking. For DOS V4, also used to support large disk partitions. This command is not available in the Epson Equity implementation of DOS.

Syntax

*dc:pathc***SHARE** */F:name_space /L:numlocks*

dc: is the disk drive holding the command.

pathc\ is the path to the command.

Switches

/F:name_space	Sets the amount of memory space (*name_space* bytes large) used for file sharing.
/L:numlocks	Sets the maximum number (*numlocks*) of file/record locks to use.

Reference

See Chapter 14.

Rules

1. If you do not give the /F switch, *name_space* is set to 2,048 bytes. Each open file uses 11 bytes plus its full file specification (disk drive name, path name, and file name). The 2,048 bytes can contain 27 files that use all 63 characters available for the full file name.
2. If you do not give the /L switch, DOS allows a default of 20 simultaneous file locks.
3. When SHARE is loaded, DOS checks for file and record locks as each file is opened, read, and written.

4. SHARE should be loaded only once after DOS has started. If you attempt to load SHARE again, DOS displays an error message.
5. SHARE normally enlarges DOS by approximately 4,900 bytes. If the number of locks (/L switch) or memory space (/F switch) is increased or decreased, DOS also increases or decreases proportionately in size.
6. The only way to remove SHARE is to restart DOS.
7. If you have not given the FCBS command in your CONFIG.SYS file, SHARE adjusts the file control block (FCB) table as if the command **FCBS = 16,8** were given.

Rules for DOS V4.0 and V4.01:

8. You must use SHARE if your hard disk has been formatted for more than 32M.
9. You can load SHARE with INSTALL in your CONFIG.SYS file.

Notes

SHARE is the DOS V3 and V4 program for file and record locking.

You use SHARE when two or more programs or processes share a single computer's files. After SHARE is loaded, DOS checks each file for locks whenever the file is opened, read, or written. If a file has been opened for exclusive use, an error message results from any subsequent attempt to open the file. If one program locks a portion of a file, an error message results if another program attempts to read or write the locked portion.

SHARE is most effective when all file-sharing programs can handle the DOS V3 and V4 functions for locking files and records. SHARE is either partially or completely ineffective with programs that do not use the DOS file- and record-locking features.

SHARE affects two or more programs running on the same computer —not two or more computers using the same file (networked computers). For networks, record and file locking are made possible by software provided with the network.

You must use SHARE if you use DOS V4.0 and V4.01 and have a hard disk formatted larger than 32M. For convenience, you can use INSTALL in the CONFIG.SYS file to activate SHARE. In the CONFIG.SYS file, for example, the line

INSTALL = C:\DOS\SHARE.EXE

activates SHARE if SHARE.EXE is found in the \DOS subdirectory on drive C:.

Messages

1. `Incorrect parameter`

   ERROR: You either gave a switch that SHARE does not recognize or put a space between the colon and the switch, or between the switch and the number. Otherwise, you probably omitted the number that goes with the switch, or you omitted the colon.

2. SHARE already loaded

   ERROR: You attempted to load SHARE a second time. The command ignores any attempt that you make to load it after it already has been loaded.

SORT *V2, V3, V4—External*

Reads lines from the standard input device, performs an ASCII sort of the lines, then writes the lines to the standard output device.

Syntax

*dc:pathc***SORT** */R* */+c*

dc: is the disk drive holding the command.

pathc\ is the path to the command.

Switches

/R Sorts in reverse order. Thus, the letter Z comes first, and the letter A comes last.

/+c Starts sorting with column number *c*.

Reference

See Chapter 9.

Rules

1. If you do not specify the /R switch, DOS sorts the file in ascending order.
2. If you do not specify the /+c switch, sorting starts with the first column (first character on the line).
3. If you do not redirect the input or output, all input is from the keyboard (standard input) and all output is to the video display (standard output). If you redirect input and output, you should use different names for the input and output files.
4. SORT uses the ASCII sequence, with the following exceptions:

   a. Control characters (such as the tab character) are not expanded.

   b. Lowercase characters are treated as uppercase characters.

   c. Some characters in the ASCII range of 128 to 225 are collated differently. SORT sorts the characters in this range in the following order:

      173-175
      156-159
      168
      131-134
      142-143

145-146
160
128
135
130
136-138
144
139
140-141
161
164-165
147-149
153
162
225
129
150-151
154
163
152
166-167
169-172
176-224

5. SORT can handle a maximum file size of 63K (64,512 characters).
6. SORT sorts text files and discards any information after and including the end-of-file marker.
7. The plus sign is mandatory when you use the /+c switch; *c* should be a positive integer.

Examples

1. **SORT <WORDS.TXT**

   SORT sorts the lines in the file WORDS.TXT and displays the sorted lines on the video screen.

2. **SORT <WORDS.TXT /R**

   SORT sorts in reverse order the lines in the file WORDS.TXT and displays the lines on the video screen.

3. **SORT /+8 <WORDS.TXT**

   SORT starts sorting at the eighth character of each line in WORDS.TXT and displays the output on the video display.

4. **DIR | SORT /+14**

   SORT displays the directory information sorted by file size. (The file size starts in the fourteenth column.) Unfortunately, other lines, such as the volume label, also are sorted starting at the fourteenth column.

Notes

SORT is useful for standard sorting, and is provided free with DOS. The program is best used in the "pipeline" with other commands, such as DIR, or for sorting simple lines of text.

SORT is a "semi-intelligent" general-purpose sorting program that has been changed significantly since DOS V2. With V3 and V4, SORT is a powerful filter, but it does have some limitations.

SORT uses the ASCII sequence with some exceptions. Look at the ASCII chart in Appendix B and note the relative positions of the numbers, punctuation marks, and letters. One of the SORT command's limitations stems from a condition shown in this chart.

SORT treats numbers as characters. Therefore, 1 always comes before 2, 3, or 9; and 0 comes before 1. The problem is that 11, 123, and 12576 all come before 2, 3, 21, or any other number that does not begin with 0 or 1.

This limitation does not affect numbers stored in strings that are left-padded with spaces (right-justified). The numbers " 1," " 121," and " 1915," (actually character strings) are sorted in the proper order. This is why Example 4, which sorts a directory by file size, works. The file size is stored as a left-padded ASCII character string.

The Ctrl-I (tab character) key combination presents another problem when the SORT command is used. When you print Ctrl-I on the screen or printer, the printer moves out to the next tab position, which is usually every eighth column. Because SORT does not expand tab characters, the output may not be what you expect. Some text editors, such as IBM Personal Editor, compress extra spaces into tab characters. Some lines appear to be out of order, therefore, when you sort a file created by Personal Editor. Loading and resaving the file without tab characters (which the Personal Editor allows) gives you the correct order when you re-sort the file.

With DOS V3 and V4, SORT handles alphabetic characters intelligently, giving upper- and lowercase characters the same treatment. SORT V2 treats these characters differently. In the ASCII characters the uppercase A precedes the uppercase Z, but the lowercase a comes after the uppercase Z. SORT V2 puts the word TEXT before the words text, TEXt, and TeXT. In V3 and V4, however, SORT treats all four of these words alike.

With V3 and V4, SORT also is intelligent in sorting certain ASCII characters in the range of 128 to 225, which includes foreign language characters and symbols. SORT V3 and V4 can handle most files containing non-American-English text.

Problems can arise when you try to sort the output from some text processors or from data files created by some languages, or when you use the I/O redirection of DOS. Like DOS, some text-processing programs fill out a file's last sector with the ASCII character null (00 hexadecimal). These null characters bubble to the top when SORT sorts a file. You will see a `^@` when you edit these files

with programs like WordStar. Fortunately, you can delete these characters with your text editor or word processor without losing information.

Some languages store data in a special compressed format. The data is not stored as ASCII text, but as compressed codes, some of which may be control characters. When you use the TYPE command to view this kind of data file, garbage appears on the screen. You should not process such a file with SORT. The command does not have the intelligence to handle these files, and the results of the sort will be of no value. When you try to sort one of these files, DOS usually responds with the message `SORT: Too many parameters`, and the command aborts.

SORT V2 has a serious problem, which was solved in SORT V3. SORT V2 sorts the Ctrl-Z, the control character used as the end-of-file marker. Many programs cannot retrieve any lines that have been placed after the end-of-file character by SORT V2. This is because most programs stop reading the file at the end-of-file marker. SORT V3, on the other hand, stops when it encounters the end-of-file marker.

Messages

1. `Invalid parameter`

   ERROR: You specified a switch that SORT does not recognize, forgot the plus sign for the column-sorting switch, or forgot to enter the number for the column-sorting switch. Check the command line and try again.

2. `SORT: Insufficient disk space`

   ERROR: SORT ran out of disk space for its temporary files or ran out of disk space when writing the output file. Free some disk space by deleting unneeded files, or move the file to a disk with more space.

   If you believe the disk has enough room for the file, run CHKDSK on the disk. If CHKDSK detects a problem, follow the directions for CHKDSK in the Command Reference.

3. `SORT: Insufficient memory`

   ERROR: You do not have enough free RAM to run SORT on this file. See the entry `Insufficient memory` under "General Messages" at the beginning of the Command Reference.

SUBST *V3.1, V3.2, V3.3, V4—External*

Creates an alias disk drive name for a subdirectory.

Syntax

To establish an alias, use the following form:

*dc:pathc***SUBST d1:** *d2:***pathname**

To delete an alias, use this form:

*dc:pathc***SUBST d1: /D**

To see the current aliases, use this form:

*dc:pathc***SUBST**

dc: is the disk drive holding the command.

pathc\ is the path to the command.

d1: is a valid disk drive name that becomes the alias or nickname. **d1:** may be a nonexistent disk drive.

*d2:***pathname** is the valid disk drive name and directory path that will be nicknamed **d1:**.

Switch

/D Deletes the alias.

Reference

See Chapter 9.

Rules

1. To display the current aliases, use **SUBST** without any disk drive name, directory name, or switch.
2. Rule 1 has one exception. You must specify **d1:**, the alias or nickname for the path. The alias must not be the current disk drive or a networked disk drive.
3. If you have not used the LASTDRIVE command in the CONFIG.SYS file, the alias can be any drive with the letter E or lower. If you have used LASTDRIVE, the alias can be any drive name up to and including the LASTDRIVE you specified.
4. *d2:* must be an existing disk drive, but not a networked disk drive or the same disk drive as **d1:**.
5. **pathname** must be a valid directory path on drive *d2:*. If you do not begin **pathname** with the backslash (\), for starting at the root directory, the current directory and the specified **pathname** are used.
6. While SUBST is in effect, you can access *d2:***pathname** by either its real name or its alias.
7. While SUBST is in effect for disk drives, take the following precautions:
   a. Be careful when using PATH, MKDIR, and RMDIR on the substituted drive. The path and subdirectories you make or remove are physically on *d2:***pathname**.
   b. Do not use the ASSIGN, BACKUP, DISKCOPY, DISKCOMP, FDISK, FORMAT, JOIN, LABEL, and RESTORE commands on a substituted disk drive.
   c. Do not use CHKDSK with the alias disk drive name.

Examples

For these examples, assume that the command LASTDRIVE=F was given in the CONFIG.SYS file, the current directory is C:\WORDS, and drive C: contains the following directory tree:

```
C:\
C:\WORDS
C:\WORDS\LETTERS
C:\WORDS\CONTRACTS
C:\BIN
C:\LANG
```

1. **SUBST E: C:\BIN**

   When you use disk drive name E:, the directory C:\BIN is actually used.

2. **SUBST F: C:LETTERS**

   Drive F: is used for the directory C:\WORDS\LETTERS. Because \WORDS is the current directory of drive C:, DOS finds LETTERS as the subdirectory of WORDS. DOS adds the \WORDS\ to the alias.

3. **SUBST**

   This command shows the current aliases. SUBST displays

```
E: ==D C:\BIN
F: ==D C:\WORDS\LETTERS
```

4. **SUBST F:**

   The alias F: is deleted. Afterward, the use of F: results in an error from DOS.

Notes

SUBST is one of the three "disk trickery" commands introduced in DOS V3.1. ASSIGN and JOIN are the other two commands.

SUBST enables you to use a disk drive name rather than a disk drive name and directory path name. SUBST also enables you to use files in subdirectories with programs that do not allow path names, such as programs written for DOS V1.

If you need several SUBST commands, use the LASTDRIVE command in your CONFIG.SYS file. Without this command, the normal LASTDRIVE is E:. On a dual-drive PC, you can use three nicknamed disks (C:, D:, and E:). On a Personal Computer AT or PC XT, you can use one or two nicknames (E: if you have two hard disks, or D: and E: if you have one hard disk drive).

While SUBST is in effect, certain DOS commands should be used with caution, others should be avoided, and some should not be used at all. When used on the alias disk, for example, MKDIR and RMDIR actually make or remove subdirectories from subdirectory *dirname*.

CHKDSK works only on the real disk drive name and path name. If you attempt to use CHKDSK with the alias disk drive name, the command displays an error message and aborts.

Avoid using ASSIGN and JOIN, because these commands work on the alias disk drive and simply increase the confusion level of DOS. It is hard to determine which physical disk drive DOS will use when ASSIGN, SUBST, and JOIN are used on the same disk drives.

Avoid using LABEL on the alias disk drive name, or you will alter the LABEL of the physical disk drive being used. If drive E: is really C:\WORDS, the **LABEL E:** command affects the volume label of C:, not the pseudodisk drive E:.

Do not use BACKUP, DISKCOPY, DISKCOMP, FDISK, FORMAT, or RESTORE with the SUBST disk drive name. These commands become confused and attempt, respectively, to back up, copy, compare, format, change, or restore a nonexistent drive.

Some of these commands display obvious error messages; others are not so obvious. The most common message is

```
Drive d: not ready
```

in which `d:` is the alias name. You should break the substitution before using these commands.

Messages

1. `Cannot SUBST a network drive`

   ERROR: You tried to use another computer's disk drive (that is, a networked disk drive) as the alias or nicknamed disk drive. You cannot use a networked disk drive with SUBST, nor can you "cover" a network disk drive with a SUBST command.

2. `Invalid parameter`

   ERROR: One of the following errors occurred: (1) the alias disk drive name exceeded the limit set by the LASTDRIVE command; (2) the alias disk drive name is the same as the current disk drive name; (3) you did not give a correct disk drive with the /D (disconnect) switch; (4) you specified a switch that SUBST does not recognize; or (5) you used the /D switch on the current disk drive.

3. `Invalid path`

   ERROR: You specified a nonexistent directory path. Make sure that the disk drive name is correct and that the path name does exist.

4. `Incorrect number of parameters`

   ERROR: You omitted a parameter when you gave the command. Perhaps you forgot to give the alias or the disk drive name and path name, or you omitted the /D switch. Check the command line and try again.

SYS *V1, V2, V3, V4—External*

Places a copy of DOS on the specified disk.

Syntax

*dc:pathc***SYS d:**

dc: is the disk drive holding the command.

pathc\ is the path to the command.

d: is the disk drive to receive the copy of DOS.

For DOS V4 you also may specify a source drive for the system files:

*dc:pathc***SYS d: d2:**

d2: is the source drive for the system files (DOS V4 only).

Reference

See Chapter 5.

Rules

1. You must name the disk drive that will receive a copy of DOS.
2. To receive the copy of DOS, the disk must be one of the following:
   a. Formatted with the /S option (DOS V1, V2, V3, or V4).
   b. Formatted with the /B option (DOS V2, V3, or V4).
   c. Formatted but empty.
   d. Formatted with a special program (non-DOS supplied) that reserves the proper disk space for the operating system.

   If you attempt to put the system on a disk that does not meet one of these conditions, a `No room for system on destination disk` message appears, and DOS does not perform the operation.
3. A copy of DOS (the IO.SYS and MSDOS.SYS files) should reside on the current disk. Otherwise, you will be prompted to insert a floppy disk containing these files into the floppy disk drive.
4. You cannot use SYS on a networked disk drive.

Notes

The SYS command places a copy of IO.SYS and MSDOS.SYS on the target disk. You also must copy COMMAND.COM onto the target disk, to enable the disk to load and execute the disk operating system.

To update a start-up disk that held a previous version of DOS, you should start the newer version of DOS and run the SYS command on the disk. Because

SYS does not copy COMMAND.COM, however, you also must copy COMMAND.COM to the disk to make it startable.

The SYS command puts DOS on application program disks sold without DOS. After processing these disks with SYS and copying COMMAND.COM to the disks, you can boot the computer system by using the application software. A floppy disk provided by the program publisher must be specially formatted, however, or SYS will not work. Check the application program's instructions to see whether you can put the system on the floppy disk.

SYS V1 and SYS for DOS V2, V3, and V4 have few differences. SYS V2 and V3 check the destination disk to see if enough space is available for DOS. SYS displays an error message if space is not available. SYS V1 places DOS on the floppy disk even if the proper space is unavailable. If you run SYS V1 on a floppy disk with existing files, some of the disk's files can turn into garbage.

MS-DOS V4 is different from earlier versions inasmuch as IO.SYS and MSDOS.SYS no longer have to be contiguous on the disk, as they once did. When you use SYS V4, SYS transfers as much of the system files to track 0 as possible, and places the rest of the files on empty space on the disk. Previous versions of SYS required that the system files be contiguous.

Messages

1. `Cannot SYS to a Network drive`

   ERROR: You cannot put the system on a networked disk drive. If the disk drive belongs to another computer, use a different disk drive. If your disk drive is being shared, pause the disk drive, perform the command and continue.

2. `Incompatible system size`

   ERROR: This disk cannot hold the hidden system files. Although the disk previously accepted IO.SYS and MSDOS.SYS, DOS cannot put the new version of the files onto the disk. These two files are larger in DOS V3 than in DOS V2 or V1. This message usually appears when you are putting the system files on DOS V1 floppy disks, but occasionally is displayed when you are placing the two files on a DOS V2 floppy disk.

   FORMAT a new floppy disk with the /S (system) switch and copy files from the old floppy disk to the new floppy disk. Be sure that you do not copy the old COMMAND.COM file.

3. `Invalid drive specification`

   ERROR: You named a nonexistent disk drive.

4. `Invalid parameter`

   ERROR: You did not name the disk drive that will hold the system files.

5. `No room for system on destination disk`

   ERROR: The disk was not formatted with the necessary reserved space for DOS. You cannot put the system on this disk.

6. `No system on default disk drive`
   `Insert system disk in drive d:`
   `and strike any key when ready`

   INFORMATION: DOS tried to load itself into memory but could not find IO.SYS, MSDOS.SYS, or COMMAND.COM. You must load these files into memory before SYS can place the operating system on a disk. Put the floppy disk that holds all three programs into drive `d:` and press a key.

7. `System transferred`

   INFORMATION: DOS successfully placed IO.SYS and MSDOS.SYS on the target disk.

TIME *V1, V2, V3, V4—Internal*

Sets and shows the system time.

Syntax

TIME *hh:mm:ss.xx*

hh is the one- or two-digit number for hours (0 to 23).

mm is the one- or two-digit number for minutes (0 to 59).

ss is the one- or two-digit number for seconds (0 to 59).

xx is the one- or two-digit number for hundredths of a second (0 to 99).

NOTE: Depending on the country code's setting in your CONFIG.SYS file, a comma may be the separator between seconds and hundredths of seconds.

Reference

See Chapter 8.

Rules

1. If you enter a valid time, DOS accepts the new time and displays no other message.

2. If you type the TIME command but do not enter the time, DOS responds:

   `Current time is hh:mm:ss.xx`
   `Enter new time`

   If DOS displays the correct time, simply press Enter, and the time is not reset. If you want to change the time setting, enter a valid time in the same format used by DOS. DOS accepts the new time without displaying another message.

3. You must enter the correct delimiters between the numbers. That is, you should place colons between the hours, minutes, and seconds, and a period

or comma between the seconds and the hundredths of a second. You must use the same decimal separator (the delimiter between seconds and hundredths of a second) as your TIME command. The United States and many other countries use only the period in this position; some countries use only the comma. Time entry and display corresponds to the country setting in your CONFIG.SYS file.

4. You do not have to enter all the information. If you change only the first parts of the time, the remaining portions are set to 0.
5. If you specify an invalid time or incorrect delimiters, DOS responds with an `Invalid time` message and prompts you to enter the time again.

Notes

TIME is used to set the computer's internal 24-hour clock. The time and date are recorded in the directory whenever you create or change a file. This information can help you find the most recent version of a file when you check your directory.

Personal computers sold in the United States use a software clock that is based on the 60 Hz power supply. This kind of clock usually loses or gains several seconds a day. The inaccuracy is not the computer's fault, but a normal problem with the AC power provided by your power company.

If you do not enter the time when starting up most PCs, the time defaults to 00:00:00.00. If you want your system to use the correct time, you must set it.

Many computers retain the time of day through a battery-backed circuit. The real-time clock is read and displayed when you start the computer. Such clocks usually are accurate to within one minute a month, which is about the same accuracy as a digital watch. After you boot your computer, DOS uses the software time clock, which suffers from the same inaccuracies as the PC clock.

Message

`Invalid time`

ERROR: You entered a nonsense time or did not punctuate the time correctly. Check your typing and try again.

TREE

V2, V3, V4—External

Displays all subdirectories on a disk or all files in each directory.

Syntax

*dc:pathc***TREE** *d: /F /A*

dc: is the disk drive holding the command.

pathc\ is the path to the command.

d: is the disk drive holding the disk you want to examine.

Switches

/F Displays all files in the directories.

/A Graphically displays the connection of subdirectories.

Reference

See Chapter 6.

Rules

1. If you do not name a disk drive, DOS examines the current disk drive.
2. All directories, starting with the root directory, are displayed with their full path names. If you use the /F switch, every file in the directory (except the . and .. entries) is listed with its full path name.

Notes

The TREE command displays all directories on a disk so that the user does not have to enter every directory, run a DIR command, and search for every <DIR> file (subdirectory).

TREE shows one directory or file name on a line and can quickly scroll off the screen. If you want a copy of the entire tree, you can redirect the output to the printer by typing the line:

TREE >PRN

or

TREE /F >PRN

The TREE command does not display the file's date, time, and size. You must use the DIR command to get this information.

When you use the /A switch, which was added with V4.0, the directory is graphically represented and subordinate directories are displayed.

Messages

1. `Directory Count Overflow`

   TREE can scan up to 3,000 subdirectories. If your hard disk has more subdirectories than this, DOS displays this message. You should reduce the number of subdirectories and use the command again.

2. `Invalid drive specification`

   ERROR: You named a nonexistent disk drive.

3. `Invalid path`

   ERROR: TREE could not use a subdirectory because something is wrong with the directory file. Run CHKDSK without the /F switch to determine what is wrong with the disk, then take the necessary corrective steps.

4. `Invalid parameter`

   ERROR: You gave a switch that TREE does not recognize, or you omitted a colon from the disk drive name.

5. `No sub-directories exist`

   INFORMATION: You have specified a disk whose root directory has no subdirectories. If you think the disk should have subdirectories, issue DIR for the disk, or run CHKDSK.

TYPE *V1, V2, V3, V4—Internal*

Displays a file's contents on-screen.

Syntax

TYPE *d:path***filename**.*ext*

d: is the disk drive holding the file that will be displayed.

path\ is the DOS path to the file.

filename.*ext* is the file to be displayed on-screen. Wild cards are not permitted.

Reference

See Chapter 7.

Rules

1. If you do not name a disk drive, DOS uses the current disk drive.
2. If you do not name a path, DOS uses the current directory.
3. The file name that you provide must match the file name in the directory.
4. All characters in the file, including control characters, are sent to the screen. Tab characters such as Ctrl-I, or CHR$(9), however, are expanded to the next eight-character boundary.
5. You cannot use the TYPE command to view a directory's contents.

Notes

The TYPE command displays a file's characters on the video screen. You can use TYPE to see a file's contents.

Strange characters appear on the screen when you use TYPE on some data files and most program files, because TYPE tries to display the machine-language instructions as ASCII characters.

The output of TYPE, like most other DOS commands, can be redirected to the printer by adding >**PRN** to the command line or by pressing Ctrl-PrtSc. (Don't forget to press Ctrl-PrtSc again to turn off the printing.)

Message

`Invalid filename or file not found`

ERROR: You have not specified a valid file name to TYPE. Check your command line and spelling to verify that a proper file name was entered.

VER *V2, V3, V4—Internal*

Shows the DOS version number on the video display.

Syntax

VER

Reference

See Chapter 8.

Note

The VER command displays a one-digit DOS version number, followed by a two-digit revision number, reminding you which DOS version (V2.0 through V4.01) the computer is using.

Messages

1. `MS-DOS Version X.XX`

   INFORMATION: You successfully invoked VER. This message indicates which version of DOS is running in the computer.

2. `Bad command or file name`

   INFORMATION: If DOS responds with this message when you attempt to invoke the VER command, you are using DOS V1.

VERIFY *V2, V3, V4—Internal*

Sets the computer to check the accuracy of data written to a disk and indicates data has been checked.

Syntax

To show the verify status, use the following form:

VERIFY

To set the verify status, use this form:

VERIFY ON

VERIFY OFF

Reference

See Chapter 7.

Rules

1. VERIFY accepts only the parameters ON or OFF.
2. Once VERIFY is on, it remains on until:
   a. A VERIFY OFF is issued.
   b. A SET VERIFY system call turns off the command.
   c. DOS is restarted.

Notes

VERIFY controls new data checking on the disk to ensure that the data have been recorded correctly. If VERIFY is off, DOS does not check the data; if VERIFY is on, DOS checks the data. VERIFY does not affect any other DOS operation.

Two factors affect the trade-off between VERIFY ON and VERIFY OFF. If VERIFY is on, data integrity is assured. If VERIFY is off, you can write to the disk faster. It usually is safe to leave VERIFY off if you are not working with critical information (such as a company's accounting figures). You are wise to turn VERIFY on, however, when you are backing up your hard disk or making important copies on floppy disks.

When verification is used, the degree of performance lost differs between hard disks and floppy disks. On average, DOS takes 68 percent more time to verify information written on a hard disk. This percentage is radically different for DOS V2, which takes only eight to nine percent more time. DOS V3 usually takes 100 percent more time when verifying information written to the floppy disk, and DOS V2 takes about 90 percent longer.

This difference between verification times for floppy and hard disks occurs partly because most hard disks have built-in routines for data verification. As a result, hard disks verify information faster than floppy disks can. Further, DOS must do all the work itself when verifying information written to a floppy disk. These factors explain the 32 percent difference in verification speed between floppy and hard disks.

The seven-fold increase in hard disk verification time between DOS V2 and V3 reflects the more thorough verification features of V3. DOS V2 does not extensively examine the information written to the hard disk, but V3 does and therefore takes more time.

Message

`Must specify ON or OFF`

ERROR: You entered a word other than **ON** or **OFF** after the VERIFY command.

VOL

V2, V3, V4—Internal

Displays the disk's volume label, if a label exists.

Syntax

VOL *d:*

d: is the disk drive whose label you want to display.

Reference

See Chapter 5.

Rule

If you do not name a disk drive, DOS uses the current disk drive.

Messages

1. `Invalid drive specification`

   ERROR: You named a nonexistent disk drive.

2. `Volume in drive d: has no label`

   INFORMATION: The disk does not have a volume label.

XCOPY

V3.2, V3.3, V4—External

Selectively copies files from one or more subdirectories.

Syntax

*dc:pathc***XCOPY** *ds:paths**filenames.exts*
*dd:pathd**filenamed.extd* */A/D/E/M/P/S/V/W*

dc: is the disk drive holding the command.

pathc\ is the path to the command.

ds: is the source disk drive, which holds the files that will be copied.

paths\ is the starting directory path to the files that will be copied.

filenames.exts is the file that will be copied. Wild cards are allowed.

dd: is the destination disk drive, which will receive the copied files. DOS refers to the destination drive as the *target*.

pathd\ is the starting directory that will receive the copied files.

filenamed.extd is the new name of the file that is copied. Wild cards are allowed.

Switches

/A Copies files whose *archive* flag is on, but does not turn off the archive flag (similar to the /M switch).

/E Creates parallel subdirectories on the destination disk even if the original subdirectory is *empty*.

/D:date Copies files that were changed or created on or after the *date* you specify. The date's form depends on the setting of the COUNTRY directive in CONFIG.SYS.

/M Copies files whose archive flag is on (*modified* files) and turns the archive flag off (similar to the /A switch).

/P Causes XCOPY to *prompt* you for approval before copying a file.

/S Copies files from this directory and all subsequent *subdirectories*.

/V *Verifies* that the copy has been recorded correctly.

/W Causes XCOPY to prompt and *wait* for the correct source floppy disk to be inserted.

Reference

See Chapter 6.

Rules

1. You must specify the source drive, path, and file name first, then the destination drive, path, and file name.
2. You cannot use a device name other than a disk drive for the source or destination name. For example, you cannot use LPT1:, COM1:, and so on.
3. The source file specification (*ds:paths\filenames.exts*) must have one or both of the following:
   a. A valid file name. Wild cards are permitted.
   b. A drive name, a path name, or both.
4. If you do not specify the source drive name, DOS uses the current drive.
5. If you do not specify the source path, DOS uses the disk drive's current directory.
6. If you specify a disk drive or path for the source but do not specify a source file name (*filenames.exts*), DOS assumes *.*.
7. If you do not give a disk drive name, DOS uses the current disk drive.
8. If you do not give a path name, DOS uses the disk's current directory.
9. If you do not give a new file name for the destination file, the copied file has the same name as the source file.
10. If a file name is given for the destination, the copied file takes the new name. Wild cards are allowed.

11. If you do not give the destination file specification, the source file specification must have either or both of the following:

    a. A disk drive name that is different from the current disk drive.

    b. A path name that is different from the current disk's current directory.

12. The source disk should not be involved in an APPEND /X command. If the source disk is part of an APPEND command, disconnect the command by using **APPEND;** before you use XCOPY.

Examples

The examples use the following sample subdirectories:

```
C:\
C:\DATA
C:\DATA\LETTERS
C:\DATA\MEMOS
C:\WP
C:\DOS
```

1. **XCOPY C:*.* A:**

   This command copies all files from the root directory of C: to the current directory of A:. In this example, XCOPY and COPY work identically.

2. **XCOPY C:*.* A: /P**

   This command is identical to Example 1, except that the /P switch is used to confirm each file individually before it is copied. The /P switch enables you to select the files that will be copied.

3. **XCOPY C:\DATA A:\ /S**

   This command first copies all files from the C:\DATA subdirectory to the root directory of A:. Because the /S switch is given, the command then copies files from subordinate subdirectories of C:\DATA to parallel subdirectories of A:. XCOPY creates the needed subdirectories on A:.

4. **XCOPY C:\DATA A:\ /S /M**

   This command is identical to Example 3, but because the /M switch is given, XCOPY copies only those files whose archive flag is on in C:\DATA and subsequent subdirectories. If all files in these directories have their archive flag on, XCOPY copies all the files. If the files in only the LETTERS subdirectory have their archive flag on, only those files are copied.

   XCOPY creates the LETTERS subdirectory on A: and copies the needed files. Because MEMOS does not contain any files to be copied (because no archive flag is on for any of its files), this subdirectory is not created.

   The /M switch, in addition to copying modified files, clears the archive flag. If the same command is issued again, the XCOPYed files are not selected until the files are modified again.

5. **XCOPY C:\DATA A:\ /S /M /E**

   This command copies the same files as in Example 4, but because the /E switch is given, XCOPY creates parallel empty subdirectories on A:\.

   Although no files are copied to the MEMOS subdirectory, the directory is created because the /E switch is used.

6. **XCOPY C:\DATA A:\ /S /A**

   This command is identical to Example 4, but because the /A switch is used, the copied files' archive flag is not changed. If the same command is given a second time, the same files are copied because their archive flag is still on.

7. **XCOPY C:\DATA A:\ /S /D:10-25-89**

   This command searches the directory C:\DATA and its subdirectories, then copies all files that were created or changed on or after October 25, 1989. The /D switch of XCOPY is identical to the /D switch of BACKUP.

8. **XCOPY A:*.* C:\DATA /P**

   This command copies all files from the root directory of A: to the directory C:\DATA. XCOPY prompts for your approval before copying the file. Press Y for yes if you want to copy the file. If you press N for no, XCOPY does not copy the current file and continues to the next file.

Notes

XCOPY is the extended COPY command. Although XCOPY runs more slowly than COPY, XCOPY offers more features and greater versatility than COPY. For example, XCOPY gives you greater control in selecting the files that will be copied. You can select the files' archive attribute and directory data and confirm the files individually before they are copied. The XCOPY command also enables you to copy through more than one directory. The command is most useful as an alternative to the DOS BACKUP command or when you want to maintain files between two computers that are in sync (two systems with hard disks or a "base" station and a laptop).

XCOPY is noticeably different from COPY and BACKUP. XCOPY cannot handle nondisk devices, for example, so you must use COPY to copy a file between serial ports or to the console. XCOPY cannot copy a single file that does not fit on one floppy disk. When the destination disk fills XCOPY stops. You should use the BACKUP command to copy a file that is larger than a floppy disk.

The /P switch enables you to use global characters in a file name that "covers" too many files, then reduce the number of files that are actually copied. When you specify the /P switch, XCOPY asks for your approval before copying each file. You can answer yes by pressing Y, or no by pressing N. The /P switch enables you to issue one XCOPY command to copy many selected files, instead of issuing several COPY commands.

The /A and /M switches are alike. Both cause XCOPY to select files that match source file names and have their archive attribute set. DOS sets the archive

attribute every time you create or modify a file. The switches can be used as a mini-backup command, or in conjunction with the BACKUP command.

The two switches act differently from each other, however, after the file is copied. /M sets the archive attribute off after the file is copied; /A does not clear the archive flag. If you use BACKUP/M to back up a file, your backup procedure can be fooled. If you use XCOPY /M on a set of files and do not change the files' archive attribute before you use BACKUP /M again, BACKUP does not back up the copied files. Your only backup copies are on the floppy disk that was created by XCOPY. If you lose the floppy disk, you have lost the only current backup of your file. For this reason, use /M with caution.

You can use the ATTRIB +A command to force XCOPY to copy files whose archive attribute has not been set. ATTRIB sets the files' archive flag. You turn on the archive flag by issuing the command

ATTRIB +A filename

If you plan to copy files from more than one subdirectory, add the /S switch to this ATTRIB command to force the archive attribute on files in subsequent directories.

If you want to use XCOPY on files that take up more space than is available on the destination disk, first make sure that the files' archive attribute is on. You can do this with the ATTRIB command. Then use the XCOPY command repeatedly with the /M or /M /S switches.

When the destination floppy disk is full, XCOPY pronounces the destination full and stops. Change floppy disks and reissue the command. The files that were copied have their archive attribute off, and XCOPY skips these files. XCOPY copies the files that have not yet been copied.

Use this technique with care if you use the BACKUP /M switch. Unless you set the archive attribute on again, BACKUP /M skips these files.

You can use the /D switch to copy files based on their directory dates. This switch is most useful for copying files that were recently created or changed. The archive attribute remains unchanged, which means that BACKUP /M backs up the right files.

The /W switch works when you want to change the source disk. XCOPY displays this message:

```
Press any key to begin copying file(s)
```

which indicates your opportunity to change diskettes. Because hard disks cannot be changed, there is little value in using the /W switch when you copy from the hard disk. Also, the switch does not affect copying to the destination disk. The proper destination disk must be in its drive before you start XCOPY.

XCOPY and APPEND /X are a troublesome combination. APPEND /X can trick XCOPY, which uses the DOS directory search functions. XCOPY either copies more files than you want or it does not copy the correct files. If you need to

use XCOPY on a disk involved in an APPEND command, disconnect APPEND before executing the XCOPY command.

Messages

1. `Access denied`

   ERROR: One of the following errors occurred: (1) you attempted to copy over a file that is marked as read-only; (2) the destination file is being used by another program or computer and is temporarily marked as read-only; or (3) the source and destination files are the same.

   If you want to copy over a file that has the read-only attribute set, use the ATTRIB command to turn off the read-only flag. If the file is being used by another program or computer, wait until the other program or computer is done, then copy the file. If the file is on another computer and you do not have permission to copy over the file, ask the operator to copy the file for you. Check the command line to make sure that the source and destination file specifications are not identical.

2. `Cannot perform a cyclic copy`

   ERROR: You used the /S switch, and one or more of the destination directories are subdirectories of the source directories. When the /S switch is given, XCOPY will not copy to destination directories that are part of the source directories. If you must copy files from more than one directory, issue individual XCOPY commands to copy the directories one at a time.

3. `Cannot XCOPY from a reserved device`

   or

   `Cannot XCOPY to a reserved device`

   ERROR: The source or destination file name is a device name. XCOPY cannot copy to or from devices. You may have misspelled one of the file names.

4. `Does %s specify a file name`
   `or directory name on the target`
   `(F = file, D = directory)?`

   INFORMATION: You gave a destination file name in which the final name does not exist as a directory. XCOPY does not know whether the final name in the destination is a file name or a directory.

   If the destination name is a directory, answer **D** for directory. XCOPY creates the needed directory and begins copying files to it. If the destination name is a file name, XCOPY copies files to this file.

5. `File cannot be copied onto itself`

   ERROR: You attempted to copy a file back to the same disk and directory with the same file name. This message usually is displayed when you misspell or omit parts of the source or destination drive, path, or file name.

Check your spelling and the source and destination names, then try the command again.

6. `File creation error`

   ERROR: XCOPY cannot create a copy of the file. Either a file of the same name exists in the destination disk and directory and is marked read-only, or the directory is full.

   See whether a file by the same name exists and whether it is read-only (use **ATTRIB filename**). If the file is marked read-only, either clear the flag by using ATTRIB or give the source file a new name when you use COPY or XCOPY.

   If you copy to the disk's root directory, the root directory may not be able to hold additional entries. Delete or move (copy then erase) files from the root directory and try the command again. If you copy to a subdirectory, run CHKDSK without the /F switch to determine whether the subdirectory has a problem.

7. `filename File not found`

   ERROR: XCOPY cannot find any files that match the source file name, `filename`. Either you named a nonexistent file or you mistyped the name.

   Find out which files already have been transferred. Determine whether the directory and file names are spelled correctly and whether the path is correct. Reissue the command.

8. `nnn File(s) copied`

   INFORMATION: XCOPY copied `nnn` files to the destination disk. This message appears regardless of any errors that may have occurred.

9. `Insufficient disk space`

   ERROR: The destination disk ran out of space. The file that was being copied when the error occurred is erased from the destination. You should either delete any unneeded files from the destination disk or use a different disk and try the command again.

10. `Invalid date`

    .ERROR: You gave an improperly phrased or nonsense date.

11. `Invalid drive specification`

    ERROR: You specified a nonexistent disk drive as the source or destination, or you forgot to place a colon after the disk drive name. Check the spelling on the command line and reissue the command.

12. `Invalid number of parameters`

    ERROR: You omitted the source and destination file name. Check the spelling and the spacing of the command line and try the command again.

13. `Invalid parameter`

    ERROR: You gave a switch that XCOPY does not recognize, or forgot to give the switch character (/) before a switch. Otherwise, you may have given too many file names. Check the spelling on the command line and try the command again.

14. `Invalid path`

    ERROR: The directory path contains an illegal character, or you entered more than the maximum 63 characters for the source or destination. If this is not the case, you probably forgot to put a space between the source and destination file names or you mistyped a name. Check the spacing and spelling on the command line and reissue the command.

15. `Lock Violation`

    ERROR: XCOPY attempted to read a file that was being used by another computer or program, and the file is locked. Try to find the computer or program that has the locked file. Try the command again when the file is free.

16. `Path not found`

    ERROR: You named a nonexistent source directory, used the wrong directory name (a directory not in the path), or mistyped a name. Issue a DIR command to make sure that you specified the correct directory. Then check the spelling on the command line and reissue the command.

17. `Path too long`

    ERROR: Either the source or destination directory path name exceeds 63 characters. A space may be missing between the source and destination file name. Check the command line. If the line is improperly phrased, try the command again with the correct phrasing. If the path has too many characters, move to the lowest possible subdirectory and try the command again with a shorter path name.

18. `Press any key to begin copying file(s)`

    INFORMATION: You used the /W switch, and XCOPY is waiting for you to press a key. If necessary, change floppy disks. Press any key for XCOPY to continue.

19. `Reading source file(s) . . .`

    INFORMATION: XCOPY is reading the source directories for file names.

20. `Sharing Violation`

    ERROR: Another computer or program is using the file that XCOPY attempted to copy. The file is marked so that XCOPY cannot read or write to the file. Wait for the other computer or program to finish with the file and try the command again.

21. `Too many open files`

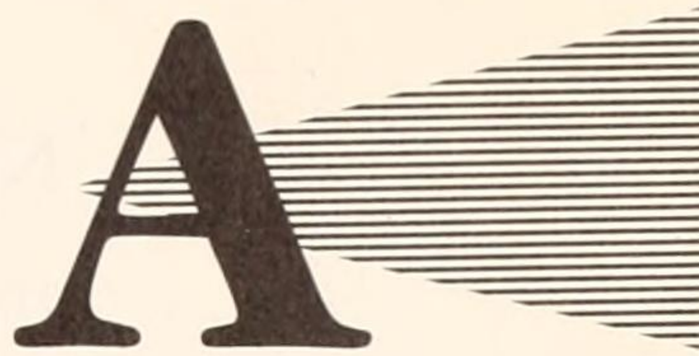
    ERROR: XCOPY tried to open the file whose name appears above this message, but was prohibited by DOS because not enough file handles were available. Another program on your computer may be using all the available file handles, or your setting of FILES= in CONFIG.SYS is too low.

    If no other program is active on your computer, edit your CONFIG.SYS file and increase the number of files given to the FILES directive. Restart your system and try the command again. If another program is active, wait for it to finish and reissue the XCOPY command.

22. `Unable to create directory`

    ERROR: XCOPY could not create a subdirectory on the destination disk because one of the following errors occurred: (1) part of the destination path name is wrong or misspelled; (2) the disk's root directory is full; (3) the disk itself is full; (4) a file by the same name as the created directory already exists; or (5) you gave a directory name that actually is a device name.

    Be sure that the destination name is correct. Check the destination disk by using the DIR command. If the disk or the root directory is full, erase files or use another destination disk. If a file exists that has the same name as the intended directory, either rename the file or change the directory's name when you reissue the XCOPY command.



Changes between Versions of DOS

Several major changes occurred between DOS V2 and V3. Additional revisions occurred between DOS V3–3.3 and between V3.3 and V4. All these changes are described briefly in this appendix.

Changes between DOS V2.x and DOS V3.0

DOS V3.0 offers several new commands, changed commands, and changed features.

New CONFIG.SYS Features

The following CONFIG.SYS directives are available in V3.0:

COUNTRY	Allows DOS to change its date, time, and other characteristics for international use
FCBS	Controls DOS's reactions to a program's use of DOS V1 file handling
LASTDRIVE	Sets the last real or nonreal disk drive that DOS will use
VDISK.SYS	Provides additional RAM (virtual) disk space

The undocumented SWITCHAR directive was dropped.

New Commands

The following commands were added to DOS V3.0:

ATTRIB	Enables the user to set the read-only attribute of a file
GRAFTABL	Allows legible display of some graphics characters if you use the Color Graphics Adapter in medium-resolution graphics mode
KEYBxx	Changes the keyboard layout for different languages
LABEL	Enables a user to add, change, or delete a disk's volume label
SELECT	Enables a user to customize the start-up disk for use with languages other than English
SHARE	Provides file sharing (file and record locking)

Changed Commands

The following commands were changed between DOS V2 and V3:

BACKUP and RESTORE	Includes the backing up of floppy disks and enables backups to be placed on another hard disk
DATE and TIME	Supports international date and time formats
FORMAT	Includes the */4* switch to format 360K floppy disks on 1.2M disk drives; also warns when a hard disk is to be formatted
GRAPHICS	Includes support for the IBM Personal Computer Compact and Color Printers

Changed Features

With DOS V3.0, a drive name and a path name can now be specified before an external command or program name. You can run programs that do not reside in the current directory or in a directory specified in the PATH command.

Changes between DOS V3.0 and V3.1

The following changes were made between DOS V3.0 and V3.1.

New Commands

The following commands are available under DOS V3.1:

JOIN	Enables the user to connect the directory structures of two different disk drives, thus creating "one" disk drive
SUBST	Enables a subdirectory to be used as though it were a disk drive

DOS V3.1 supports the IBM PC Network commands.

Changed Commands

The following commands were changed in V3.1:

LABEL	Prompts the user before deleting a volume label
SHELL	Includes the */E:size* switch; size in 16-byte paragraphs
TREE	Displays files in the root directory when the */F* switch is used

Changes between DOS V3.1 and V3.2

The following changes were made between DOS V3.1 and V3.2.

New CONFIG.SYS Features

STACKS	Sets the number and size of the DOS internal stacks

The DRIVER.SYS device driver was added to support various-sized floppy disks, particularly 720K microfloppy drives on Personal Computers.

New Feature

Support for the IBM Token Ring was added.

New Commands

REPLACE	Selectively updates files in one or many directories; adds missing files to a directory
XCOPY	Copies files from one or more directories to another; selectively copies files

Changed Commands/Directives

ATTRIB	*+A/–A* switch added; controls the archive attribute of files
COMMAND	*/E* switch added; supports the environment size
DISKCOPY/DISKCOMP	Supports 720K floppy disks
FORMAT	Supports formatting of 720K floppy disks; requests verification before formatting a nonremovable disk that has a volume label; disk drive name required
SELECT	Formats the hard disk and copies DOS files
SHELL	*/E:size* switch specifies the environment size in bytes, not 16-byte paragraphs

Changes between DOS V3.2 and V3.3

The following changes were made between DOS V3.2 and V3.3.

New CONFIG.SYS Options

The following device drivers for use with the DEVICE directive were added:

DISPLAY.SYS	Supports code pages (multiple fonts) on EGA, VGA, and PC Convertible displays
PRINTER.SYS	Supports code pages (multiple fonts) on the IBM ProPrinter and Quietwriter III printers

New Features

Support was added for 1.44M microfloppy disks, for COM4: and 19,200 baud rates, and for switchable code pages (international character fonts).

New Commands

The following commands were added to DOS V3.3:

APPEND	A PATH-like command for data files
CHCP	Provides code page changing
FASTOPEN	Provides a directory-caching program for hard disks
NLSFUNC	Provides support for additional character sets (code pages) and for country-dependent information

Changed Commands/Directives

The following commands and CONFIG.SYS directives were changed for DOS V3.3:

ATTRIB	Has the */S* switch to change the attributes of files in subdirectories
BACKUP	Has the */F* switch to format floppy disks, the */T* switch to back up files based on their time, and the */L* switch to produce a log file; also places all backed-up files into a single file
batch files	Adds support for using the environment variable (%variable%), @ for suppressing display of a line, and the CALL subcommand for running a second batch file and returning control to the first batch file
BUFFERS	Default buffers based on random-access memory in the computer
COUNTRY	Adds support for code pages and a separate country information file (COUNTRY.SYS)
DATE and TIME	Sets the computer's clock/calendar
DISKCOPY/DISKCOMP	Supports 1.44M floppy disks

FDISK	Supports multiple logical disks on a large hard disk
FORMAT	Adds the */N* switch for number of sectors and the */T* switch for number of tracks
GRAFTABL	Supports code pages; also supports additional devices and higher baud rates
KEYB	Replaces the KEYBxx programs and supports additional layouts
MODE	Supports code pages; also supports additional devices and higher baud rate
RESTORE	Adds the */N* switch to restore erased or modified files, the */B* switch to restore files modified before a given date, and the */L* and */E* switches to restore files modified after or before a given time
SHELL	Default environment size changed from 128 to 160 bytes

Changes between DOS V3.3 and V4

The following changes were made between DOS V3.3 and DOS V4.

New CONFIG.SYS Features

INSTALL	Allows loading of terminate-and-stay-resident programs that were previously loaded from the DOS command prompt or in the AUTOEXEC.BAT file; includes FASTOPEN.EXE, KEYB.COM, NLSFUNC.EXE, and SHARE.EXE
REM	Allows the insertion of remarks in a CONFIG.SYS file, which DOS will ignore when the computer boots
SWITCHES	Disables Enhanced Keyboard functions so that software that cannot use those functions will remain compatible
XMA2EMS.SYS	A new device driver for expanded memory
XMAEM.SYS	Allows emulation of an expanded memory adapter on 80386 machines

New Features

A new user interface, the DOS Shell, allows the running of programs and the management of files using a graphics- or text-oriented menu system. Many error messages were changed, and error checking was refined.

New Commands

MEM	Provides a report on available conventional, extended, and expanded memory, and lists how much of each is unused
TRUENAME	Lists the actual name of a drive or directory affected by a JOIN or SUBST command

Changed Commands/Directives

The following commands and CONFIG.SYS directives were changed between DOS V3.3 and V4:

ANSI.SYS	Gained three new parameters: */X*, which redefines keys added to Enhanced Keyboards; */L*, which tells DOS to override any applications program that resets the number of screen rows to 25; and */K*, which turns off extended keyboard functions to comply with noncompatible software
APPEND	Can ignore file operations that already include a drive or path in the original specification
BACKUP	Automatically formats target floppy disks if necessary
BUFFERS	Allows the */X* switch, which tells DOS to use expanded memory; specify up to 10,000 buffers and 1 to 8 *look-ahead* buffers
CHKDSK	Shows the disk's serial number and tells the size and number of allocation units
COUNTRY	Provides support for Japanese, Korean, and Chinese characters—on special Asian hardware only
DEL/ERASE	Includes */P*, which allows you to verify each file name before it is deleted

DIR	Shows the disk's serial number
DISPLAY.SYS	Checks hardware and automatically chooses the most appropriate type of active display if you don't specify an adapter type
FASTOPEN	May be loaded from CONFIG.SYS, and has the */X* parameter, which tells DOS to use expanded memory
FDISK	Supports larger disk partitions and has easier-to-use menus and displays
FORMAT	Has a */V:label* parameter, which lets you specify the label on the command line as you start to format a disk, and a new */F:size* parameter, which allows you to indicate the size of a floppy disk when you want to format the disk for less than its maximum capacity
GRAFTABL	Supports code page 850
GRAPHICS	Supports EGA and VGA adapters and can support more printers
KEYB	Has a */ID.nnn* parameter, which can be used to choose a specific keyboard for countries like France, Italy, and Great Britain that have more than one Enhanced Keyboard
MODE	Lets you specify the keyboard rate and number of lines displayed on the screen; has parameters for the COM ports
PRINTER.SYS	Has enhanced support for the IBM ProPrinter
REPLACE	Has the */U* parameter, which tells DOS to update files that have a date and time more recent than that of a file with the same name on the target disk
SELECT	Enhanced to install DOS
SYS	Changed to allow specification of an optional source drive
TIME	Allows either a 12-hour or 24-hour clock, depending on the country code in use
TREE	Includes graphics and indentations to show various subdirectory levels

VDISK.SYS	Has the */X* parameter, which tells DOS to use expanded memory, and the */E* parameter, which tells DOS to use extended memory

B

ASCII and Extended ASCII Codes

This appendix presents the ASCII, Extended ASCII, and Extended-Function ASCII codes. In the tables, a ^ represents the Control (Ctrl) key. For example, ^C represents Ctrl-C.

ASCII Codes

The codes for the American Standard Code for Information Interchange (ASCII) are presented in the following table.

Decimal	*Hex*	*Octal*	*Binary*	*Graphic Character*	*ASCII Meaning*
0	0	0	00000000		^@ NUL (null)
1	1	1	00000001	☺	^A SOH (start-of-header)
2	2	2	00000010	●	^B STX (start-of-transmission)
3	3	3	00000011	♥	^C ETX (end-of-transmission)
4	4	4	00000100	♦	^D EOT (end-of-text)
5	5	5	00000101	♣	^E ENQ (enquiry)
6	6	6	00000110	♠	^F ACK (acknowledge)
7	7	7	00000111	•	^G BEL (bell)
8	8	10	00001000	◘	^H BS (backspace)
9	9	11	00001001	○	^I HT (horizontal tab)
10	A	12	00001010	◙	^J LF (line feed - also ^Enter)
11	B	13	00001011	♂	^K VT (vertical tab)
12	C	14	00001100	♀	^L FF (form feed)
13	D	15	00001101	♪	^M CR (carriage return)
14	E	16	00001110	♫	^N SO
15	F	17	00001111	☼	^O SI
16	10	20	00010000	►	^P DLE
17	11	21	00010001	◄	^Q DC1
18	12	22	00010010	↕	^R DC2

Decimal	*Hex*	*Octal*	*Binary*	*Graphic Character*	*ASCII Meaning*
19	13	23	00010011	‼	^S DC3
20	14	24	00010100	¶	^T DC4
21	15	25	00010101	§	^U NAK
22	16	26	00010110	▬	^V SYN
23	17	27	00010111	↨	^W ETB
24	18	30	00011000	↑	^X CAN (cancel)
25	19	31	00011001	↓	^Y EM
26	1A	32	00011010	→	^Z SUB (also end-of-file)
27	1B	33	00011011	←	^[ESC (Escape)
28	1C	34	00011100	∟	^\ FS (field separator)
29	1D	35	00011101	↔	^] GS
30	1E	36	00011110	▲	^^ RS (record separator)
31	1F	37	00011111	▼	^_ US
32	20	40	00100000		Space
33	21	41	00100001	!	!
34	22	42	00100010	"	"
35	23	43	00100011	#	#
36	24	44	00100100	$	$
37	25	45	00100101	%	%
38	26	46	00100110	&	&
39	27	47	00100111	'	'
40	28	50	00101000	(	(
41	29	51	00101001	)	)
42	2A	52	00101010	*	*
43	2B	53	00101011	+	+
44	2C	54	00101100	,	,
45	2D	55	00101101	-	-
46	2E	56	00101110	.	.
47	2F	57	00101111	/	/
48	30	60	00110000	0	0
49	31	61	00110001	1	1
50	32	62	00110010	2	2
51	33	63	00110011	3	3
52	34	64	00110100	4	4
53	35	65	00110101	5	5
54	36	66	00110110	6	6
55	37	67	00110111	7	7
56	38	70	00111000	8	8
57	39	71	00111001	9	9
58	3A	72	00111010	:	:
59	3B	73	00111011	;	;
60	3C	74	00111100	<	<
61	3D	75	00111101	=	=
62	3E	76	00111110	>	>
63	3F	77	00111111	?	?
64	40	100	01000000	@	@
65	41	101	01000001	A	A
66	42	102	01000010	B	B
67	43	103	01000011	C	C
68	44	104	01000100	D	D

Decimal	*Hex*	*Octal*	*Binary*	*Graphic Character*	*ASCII Meaning*
69	45	105	01000101	E	E
70	46	106	01000110	F	F
71	47	107	01000111	G	G
72	48	110	01001000	H	H
73	49	111	01001001	I	I
74	4A	112	01001010	J	J
75	4B	113	01001011	K	K
76	4C	114	01001100	L	L
77	4D	115	01001101	M	M
78	4E	116	01001110	N	N
79	4F	117	01001111	O	O
80	50	120	01010000	P	P
81	51	121	01010001	Q	Q
82	52	122	01010010	R	R
83	53	123	01010011	S	S
84	54	124	01010100	T	T
85	55	125	01010101	U	U
86	56	126	01010110	V	V
87	57	127	01010111	W	W
88	58	130	01011000	X	X
89	59	131	01011001	Y	Y
90	5A	132	01011010	Z	Z
91	5B	133	01011011	[	[
92	5C	134	01011100	\	\
93	5D	135	01011101	]	]
94	5E	136	01011110	^	^
95	5F	137	01011111	_	_
96	60	140	01100000	`	`
97	61	141	01100001	a	a
98	62	142	01100010	b	b
99	63	143	01100011	c	c
100	64	144	01100100	d	d
101	65	145	01100101	e	e
102	66	146	01100110	f	f
103	67	147	01100111	g	g
104	68	150	01101000	h	h
105	69	151	01101001	i	i
106	6A	152	01101010	j	j
107	6B	153	01101011	k	k
108	6C	154	01101100	l	l
109	6D	155	01101101	m	m
110	6E	156	01101110	n	n
111	6F	157	01101111	o	o
112	70	160	01110000	p	p
113	71	161	01110001	q	q
114	72	162	01110010	r	r
115	73	163	01110011	s	s
116	74	164	01110100	t	t
117	75	165	01110101	u	u
118	76	166	01110110	v	v

Decimal	*Hex*	*Octal*	*Binary*	*Graphic Character*	
119	77	167	01110111	w	w
120	78	170	01111000	x	x
121	79	171	01111001	y	y
122	7A	172	01111010	z	z
123	7B	173	01111011	{	{
124	7C	174	01111100	|	|
125	7D	175	01111101	}	}
126	7E	176	01111110	~	~
127	7F	177	01111111	Δ	Del
128	80	200	10000000	Ç	
129	81	201	10000001	ü	
130	82	202	10000010	é	
131	83	203	10000011	â	
132	84	204	10000100	ä	
133	85	205	10000101	à	
134	86	206	10000110	ȧ	
135	87	207	10000111	ç	
136	88	210	10001000	ê	
137	89	211	10001001	ë	
138	8A	212	10001010	è	
139	8B	213	10001011	ï	
140	8C	214	10001100	î	
141	8D	215	10001101	ì	
142	8E	216	10001110	Ä	
143	8F	217	10001111	Ȧ	
144	90	220	10010000	É	
145	91	221	10010001	æ	
146	92	222	10010010	Æ	
147	93	223	10010011	ô	
148	94	224	10010100	ö	
149	95	225	10010101	ò	
150	96	226	10010110	û	
151	97	227	10010111	ù	
152	98	230	10011000	ÿ	
153	99	231	10011001	Ö	
154	9A	232	10011010	Ü	
155	9B	233	10011011	¢	
156	9C	234	10011100	£	
157	9D	235	10011101	¥	
158	9E	236	10011110	Pt	
159	9F	237	10011111	ƒ	
160	A0	240	10100000	á	
161	A1	241	10100001	í	
162	A2	242	10100010	ó	
163	A3	243	10100011	ú	
164	A4	244	10100100	ñ	
165	A5	245	10100101	Ñ	
166	A6	246	10100110	ª	
167	A7	247	10100111	º	
168	A8	250	10101000	¿	
169	A9	251	10101001	⌐	

Decimal	*Hex*	*Octal*	*Binary*	*Graphic Character*
170	AA	252	10101010	¬
171	AB	253	10101011	½
172	AC	254	10101100	¼
173	AD	255	10101101	¡
174	AE	256	10101110	«
175	AF	257	10101111	»
176	B0	260	10110000	▒
177	B1	261	10110001	▒
178	B2	262	10110010	▓
179	B3	263	10110011	│
180	B4	264	10110100	┤
181	B5	265	10110101	╡
182	B6	266	10110110	╢
183	B7	267	10110111	╖
184	B8	270	10111000	╕
185	B9	271	10111001	╣
186	BA	272	10111010	║
187	BB	273	10111011	╗
188	BC	274	10111100	╝
189	BD	275	10111101	╜
190	BE	276	10111110	╛
191	BF	277	10111111	┐
192	C0	300	11000000	└
193	C1	301	11000001	┴
194	C2	302	11000010	┬
195	C3	303	11000011	├
196	C4	304	11000100	─
197	C5	305	11000101	┼
198	C6	306	11000110	╞
199	C7	307	11000111	╟
200	C8	310	11001000	╚
201	C9	311	11001001	╔
202	CA	312	11001010	╩
203	CB	313	11001011	╦
204	CC	314	11001100	╠
205	CD	315	11001101	═
206	CE	316	11001110	╬
207	CF	317	11001111	╧
208	D0	320	11010000	╨
209	D1	321	11010001	╤
210	D2	322	11010010	╥
211	D3	323	11010011	╙
212	D4	324	11010100	╘
213	D5	325	11010101	╒
214	D6	326	11010110	╓
215	D7	327	11010111	╫
216	D8	330	11011000	╪
217	D9	331	11011001	┘
218	DA	332	11011010	┌
219	DB	333	11011011	▒

Decimal	*Hex*	*Octal*	*Binary*	*Graphic Character*
220	DC	334	11011100	▄
221	DD	335	11011101	▌
222	DE	336	11011110	▐
223	DF	337	11011111	▀
224	E0	340	11100000	α
225	E1	341	11100001	ß
226	E2	342	11100010	Γ
227	E3	343	11100011	π
228	E4	344	11100100	Σ
229	E5	345	11100101	σ
230	E6	346	11100110	µ
231	E7	347	11100111	τ
232	E8	350	11101000	Φ
233	E9	351	11101001	Θ
234	EA	352	11101010	Ω
235	EB	353	11101011	δ
236	EC	354	11101100	∞
237	ED	355	11101101	φ
238	EE	356	11101110	∈
239	EF	357	11101111	∩
240	F0	360	11110000	≡
241	F1	361	11110001	±
242	F2	362	11110010	≥
243	F3	363	11110011	≤
244	F4	364	11110100	⌠
245	F5	365	11110101	⌡
246	F6	366	11110110	÷
247	F7	367	11110111	≈
248	F8	370	11111000	°
249	F9	371	11111001	∙
250	FA	372	11111010	·
251	FB	373	11111011	√
252	FC	374	11111100	n
253	FD	375	11111101	²
254	FE	376	11111110	■
255	FF	377	11111111	

Extended ASCII Keyboard Codes

Certain keys cannot be represented by the standard ASCII codes. To represent the codes, a two-character sequence is used. The first character is always an ASCII NUL (0). The second character and its translation are listed in the following table. Some codes expand to multi-keystroke characters.

If an asterisk (*) appears in the column *Enhanced Only*, the sequence is available only on the Enhanced Keyboards (101- and 102-key keyboards).

Enhanced Only	*Decimal*	*Hex*	*Octal*	*Binary*	*Extended ASCII Meaning*
*	1	01	001	00000001	Alt-Esc
	3	03	003	00000011	Null (null character)
*	14	0E	016	00001110	Alt-Backspace
	15	0F	017	00001111	Shift-Tab (back-tab)
	16	10	020	00010000	Alt-Q
	17	11	021	00010001	Alt-W
	18	12	022	00010010	Alt-E
	19	13	023	00010011	Alt-R
	20	14	024	00010100	Alt-T
	21	15	025	00010101	Alt-Y
	22	16	026	00010110	Alt-U
	23	17	027	00010111	Alt-I
	24	18	030	00011000	Alt-O
	25	19	031	00011001	Alt-P
*	26	1A	032	00011010	Alt-[
*	27	1B	033	00011011	Alt-]
*	28	1C	034	00011100	Alt-Enter
	30	1E	036	00011110	Alt-A
	31	1F	037	00011111	Alt-S
	32	20	040	00100000	Alt-D
	33	21	041	00100001	Alt-F
	34	22	042	00100010	Alt-G
	35	23	043	00100011	Alt-H
	36	24	044	00100100	Alt-J
	37	25	045	00100101	Alt-K
	38	26	046	00100110	Alt-L
*	39	27	047	00100111	Alt-;
*	40	28	050	00101000	Alt-'
*	41	29	051	00101001	Alt-`
*	43	2B	053	00101011	Alt-\
	44	2C	054	00101100	Alt-Z
	45	2D	055	00101101	Alt-X
	46	2E	056	00101110	Alt-C
	47	2F	057	00101111	Alt-V
	48	30	060	00110000	Alt-B
	49	31	061	00110001	Alt-N
	50	32	062	00110010	Alt-M
*	51	33	063	00110011	Alt-,
*	52	34	064	00110100	Alt-.
*	53	35	065	00110101	Alt-/

*	55	37	067	00110111	Alt-* (keypad)
	57	39	071	00111001	Alt-space bar
	59	3B	073	00111011	F1
	60	3C	074	00111100	F2
	61	3D	075	00111101	F3
	62	3E	076	00111110	F4
	63	3F	077	00111111	F5
	64	40	100	01000000	F6
	65	41	101	01000001	F7
	66	42	102	01000010	F8
	67	43	103	01000011	F9
	68	44	104	01000100	F10
	71	47	107	01000111	Home
	72	48	110	01001000	↑
	73	49	111	01001001	PgUp
	74	4A	112	01001010	Alt-− (keypad)
	75	4B	113	01001011	←
	76	4C	114	01001100	Shift-5 (keypad)
	77	4D	115	01001101	→
	78	4E	116	01001110	Alt-+ (keypad)
	79	4F	117	01001111	End
*	80	50	120	01010000	↓
*	81	51	121	01010001	PgDn
*	82	52	122	01010010	Ins (Insert)
	83	53	123	01010011	Del (Delete)
	84	54	124	01010100	Shift-F1
	85	55	125	01010101	Shift-F2
	86	56	126	01010110	Shift-F3
	87	57	127	01010111	Shift-F4
	88	58	130	01011000	Shift-F5
	89	59	131	01011001	Shift-F6
	90	5A	132	01011010	Shift-F7
	91	5B	133	01011011	Shift-F8
	92	5C	134	01011100	Shift-F9
	93	5D	135	01011101	Shift-F10
	94	5E	136	01011110	Ctrl-F1
	95	5F	137	01011111	Ctrl-F2
	96	60	140	01100000	Ctrl-F3
	97	61	141	01100001	Ctrl-F4
	98	62	142	01100010	Ctrl-F5
	99	63	143	01100011	Ctrl-F6
	100	64	144	01100100	Ctrl-F7
	101	65	145	01100101	Ctrl-F8
	102	66	146	01100110	Ctrl-F9

	103	67	147	01100111	Ctrl-F10
	104	68	150	01101000	Alt-F1
	105	69	151	01101001	Alt-F2
	106	6A	152	01101010	Alt-F3
	107	6B	153	01101011	Alt-F4
	108	6C	154	01101100	Alt-F5
	109	6D	155	01101101	Alt-F6
	110	6E	156	01101110	Alt-F7
	111	6F	157	01101111	Alt-F8
	112	70	160	01110000	Alt-F9
	113	71	161	01110001	Alt-F10
	114	72	162	01110010	Ctrl-PrtSc
	115	73	163	01110011	Ctrl-←
	116	74	164	01110100	Ctrl-→
	117	75	165	01110101	Ctrl-End
	118	76	166	01110110	Ctrl-PgDn
	119	77	167	01110111	Ctrl-Home
	120	78	170	01111000	Alt-1 (keyboard)
	121	79	171	01111001	Alt-2 (keyboard)
	122	7A	172	01111010	Alt-3 (keyboard)
	123	7B	173	01111011	Alt-4 (keyboard)
	124	7C	174	01111100	Alt-5 (keyboard)
	125	7D	175	01111101	Alt-6 (keyboard)
	126	7E	176	01111110	Alt-7 (keyboard)
	127	7F	177	01111111	Alt-8 (keyboard)
	128	80	200	10000000	Alt-9 (keyboard)
	129	81	201	10000001	Alt-0 (keyboard)
	130	82	202	10000010	Alt-− (keyboard)
	131	83	203	10000011	Alt-= (keyboard)
	132	84	204	10000100	Ctrl-PgUp
*	133	85	205	10000101	F11
*	134	86	206	10000110	F12
*	135	87	207	10000111	Shift-F11
*	136	88	210	10001000	Shift-F12
*	137	89	211	10001001	Ctrl-F11
*	138	8A	212	10001010	Ctrl-F12
*	139	8B	213	10001011	Alt-F11
*	140	8C	214	10001100	Alt-F12
	141	8D	215	10001101	Ctrl-↑/8 (keypad)
	142	8E	216	10001110	Ctrl-− (keypad)
	143	8F	217	10001111	Ctrl-5 (keypad)
	144	90	220	10010000	Ctrl-+ (keypad)
	145	91	221	10010001	Ctrl-↓/2 (keypad)
	146	92	222	10010010	Ctrl-Ins/0 (keypad)

	147	93	223	10010011	Ctrl-Del/. (keypad)
	148	94	224	10010100	Ctrl-Tab
*	149	95	225	10010101	Ctrl-/ (keypad)
*	150	96	226	10010110	Ctrl-* (keypad)
*	151	97	227	10010111	Alt-Home
*	152	98	230	10011000	Alt-↕
*	153	99	231	10011001	Alt-Page Up
*	155	9B	233	10011011	Alt-←
*	157	9D	235	10011101	Alt-→
*	159	9F	237	10011111	Alt-End
*	160	A0	240	10100000	Alt-↑
*	161	A1	241	10100001	Alt-Page Down
*	162	A2	242	10100010	Alt-Insert
*	163	A3	243	10100011	Alt-Delete
*	164	A4	244	10100100	Alt-/ (keypad)
*	165	A5	245	10100101	Alt-Tab
*	166	A6	256	10100110	Alt-Enter (keypad)

Extended-Function ASCII Codes

The following extended codes are available only with Enhanced Keyboards (101- and 102-key keyboards); they are available for key reassignment only under DOS V4. The keys include the six-key editing pad and the four-key cursor-control pad. To reassign these keys, you must give the DEVICE = ANSI.SYS /X directive or the enable-extended-function-codes escape sequence (Esc[1q). All extended codes are prefixed by 224 decimal (E0 hex).

Enhanced Function	*Decimal*	*Hex*	*Octal*	*Binary*	*Extended ASCII Meaning*
*	71	47	107	01000111	Home
*	72	48	110	01001000	↑
*	73	49	111	01001001	Page Up
*	75	4B	113	01001011	←
*	77	4D	115	01001101	→
*	79	4F	117	01001111	End
*	80	50	120	01010000	↓
*	81	51	121	01010001	Page Down
*	82	52	122	01010010	Insert
*	83	53	123	01010011	Delete
*	115	73	163	01110011	Ctrl-←
*	116	74	164	01110100	Ctrl-→
*	117	75	165	01110101	Ctrl-End
*	118	76	166	01110110	Ctrl-Page Down

*	119	77	167	01110111	Ctrl-Home
*	132	84	204	10000100	Ctrl-Page Up
*	141	8D	215	10001101	Ctrl-↑
*	145	91	221	10010001	Ctrl-↓
*	146	92	222	10010010	Ctrl-Insert
*	147	93	223	10010011	Ctrl-Delete

DOS Control and Editing Keys

Control Keys

Enter, Return, or ←	Tells DOS to act on the line you just typed
←	Backs up and deletes one character from the line
Ctrl-C Ctrl-Break	Stops a command
Ctrl-Num Lock Ctrl-S	Freezes the video display; pressing any other key restarts the display
PrtSc Print Screen	Prints the contents of the video display (print-screen feature)
Ctrl-PrtSc Ctrl-Print Screen Ctrl-P	Echoes lines sent to the screen to the printer also; giving this sequence a second time turns off this function (printer echo feature)
Ctrl-Alt-Del	Restarts DOS (system reset)

Editing Keys

When you type a line and press Enter, DOS copies the line into an input buffer. By using certain keys, you can use the same line over and over again. The following keys enable you to edit the input buffer line. When you press Enter, the new line is placed into the primary input buffer as DOS executes the line.

|← →|	Moves to the next tab stop

Esc	Cancels the current line and does not change buffer
Ins	Enables you to insert characters in the line
Del	Deletes a character from the line
F1 or →	Copies one character from the previous command line
F2	Copies all characters from the previous command line up to, but not including, the next character you type
F3	Copies all remaining characters from the previous command line
F4	Deletes all characters from the previous command line up to, but not including, the next character you type (opposite of F2)
F5	Moves the current line you are typing into the buffer but does not allow DOS to execute the line
F6	Produces an end-of-file marker when you copy from the console to a disk file

Special Key Definitions within the DOS Shell

DOS V4 assigns special functions to some of the keys used with the Start Programs menu.

Enter or ←	Places the information you select or type into memory
Esc	Cancels the current function
F1	Displays a series of text screens with help information
PgUp PgDn	Scrolls through "pages" of information
F2	Saves the information typed in an entry field and completes the copy operation in Start Programs

F3	Returns you to the DOS command prompt from Start Programs and the File System; cancels copying operations in Start Programs
F4	Creates the || marker that separates the commands used to start a program
F9	Displays key assignments within HELP; returns an entry field to its original value; switches the View function between ASCII and hexadecimal format
Shift-F9	Takes you to the DOS command prompt; typing **EXIT** returns you to the DOS Shell
F10	Moves the cursor to the action bar if the cursor is located in the Main Group or to the Main Group if the cursor is located in the action bar
Space bar	Selects or deselects a file name
Tab	Moves the selection cursor to different areas of the File System screen

Key Assignments within the DOS Shell's HELP Feature

DOS V4 provides a context-sensitive help feature for all selectable items and entry fields. Within HELP, special function key assignments assist you in using HELP.

Esc	Returns you to the menu
F1	Provides instructions on how to use HELP
F9	Lists the special key definitions in HELP
F11 or Alt-F1	Displays an index of available HELP screens

D

Preparing Your Hard Disk

IMPORTANT: *Some dealers install the DOS operating system and applications programs on your hard disk before you receive your computer. If only DOS has been installed, you may skip the steps in these first two sections, which describe two of the processes for setting up the hard disk. If, however, you choose to complete the steps in these sections, you must complete both sections before proceeding to the final section.*

If your dealer has installed some applications programs, such as an accounting package, a word processor, or a data management program, you should not complete the steps outlined in the first and second sections of this appendix.

Before you can use your hard disk, you must perform three distinct processes: (1) assign the disk one or more partitions by using FDISK, (2) format the hard disk by using FORMAT, and (3) set up the hard disk for daily operations using several DOS commands. Before begining these processes, you must have DOS up and running on your computer.

If you have an IBM Personal Computer AT, you must run the SETUP program before starting the steps outlined in the sections of this appendix. The information on SETUP is located in the Installation and Setup manual for your Personal Computer AT. In the setup procedure, you will be asked for the disk drive type. Currently, all IBM 20-megabyte disk drives are type 2, and all IBM 30-megabyte disk drives are type 20. If you are unsure of your disk drive type, remove the cover of your AT and read the large number on the front of the disk drive.

If you have an IBM PS/2 Micro Channel computer (Model 50, 50Z, 60, 70, or 80), you should boot the system from the Reference Disk. Follow the instructions, and the system will automatically configure itself for the hard disk and adapter cards you have installed.

FDISK is slightly different for DOS V3 and DOS V4. Each is addressed in separate sections in this appendix.

Using FDISK with DOS V3.2 and Earlier Versions

FDISK works a little differently with DOS V3.2 and earlier versions than it does with DOS V3.3. The differences are noted as they occur in the following instructions.

The hard disk of your computer can be used by more than one disk operating system. To accommodate the people who use more than one operating system, IBM provides the FDISK program, which divides the hard disk into partitions. Each partition is used with a different operating system.

Most people use only DOS. The instructions given in this first section are for setting up your hard disk for a single DOS partition.

Step 1. *Insert your DOS start-up disk (one of the disks that came with your DOS program) into the first disk drive.*

This drive will be the left or top disk drive. Close the disk drive door. Start your computer by either turning on the power if the computer is off or pressing Ctrl-Alt-Del if the computer is on. (Hold down the Ctrl and Alt keys and then press the Del key.)

Step 2. *If you are not using a Personal Computer AT, type the correct date and press Enter. Separate the month, day, and year with hyphens.*

If you are using an AT, check the date. If the date is incorrect, enter the correct date and then press Enter. Remember that the date was incorrect.

Step 3. *If you are not using an AT, enter the correct time.*

Because DOS uses a universal, 24-hour clock, add 12 to afternoon and evening hours. Separate the hours, minutes, and seconds with colons. You don't need to enter the hundredths of seconds.

If you are using an AT, check the time. If the time is incorrect, enter the correct time and press Enter. If the time is off by more than a minute or two, remember this fact.

You should now see the DOS prompt:

```
A>
```

Step 4. *Type FDISK and press Enter.*

The screen should look something like the following:

```
IBM Personal Computer
Fixed Disk Setup Program Version 3.20
(C)Copyright IBM Corp. 1983, 1985
```

```
FDISK Options

Current Fixed Disk Drive: 1

Choose one of the following:

        1. Create DOS Partition
        2. Change Active Partition
        3. Delete DOS Partition
        4. Display Partition Data
        5. Select Next Fixed Disk Drive

Enter choice: [1]
```

Step 5. *Choose option 1 by pressing Enter.*

For DOS V3.2 and earlier, if the following message appears, a DOS partition has already been created on your hard disk:

```
DOS partition already exists
```

You should skip this section and the next one on formatting your disk. Press the Esc key to return to DOS.

If this message does not appear, and you have DOS V3.2 or an earlier version, your screen should look something like this:

```
Create DOS Partition

Current Fixed Disk Drive: 1

Do you wish to use the entire fixed

disk for DOS (Y/N).............? [Y]
```

Step 6. *Press Enter to inform FDISK that DOS should use the entire hard disk.*

The red light for the hard disk should go on briefly, and the following message should appear:

```
System will now restart

Insert DOS disk in drive A:

Press any key when ready...
```

Step 7. *Press any key.*

Because the DOS disk is already in drive A:, DOS will restart. Answer the date and time questions again (steps 2 and 3) and then proceed to the next section on formatting the hard disk.

Using FDISK with DOS V3.3 and V4

***Step 1.** After you have booted DOS, load the disk with FDISK in drive A.*

For DOS V3.3 this will be the Operating disk; for DOS V4, the Install disk. At the A> prompt, type

FDISK

You'll see a screen like this one:

```
                IBM DOS Version 4.00
                 Fixed Disk Setup Program
              (C) Copyright IBM Corp. 1983, 1988

                        FDISK Options

Current fixed disk drive:1

Choose one of the following:

     1. Create DOS Partition or Logical DOS Drive
     2. Set active partition
     3. Delete DOS Partition or Logical DOS Drive
     4. Display partition information
     5. Select next fixed disk drive

   Enter choice:[ ]

   Press Esc to exit FDISK
```

For DOS V3.3, option 2 in this display reads `Change Active Partition`, and option 4 reads `Display Partition Date`. DOS V3.3 allows you to have more than one bootable partition, so it is possible to use FDISK to change the active (bootable) partition from one to another. DOS V4 allows you only to activate the primary DOS partition.

With either version of DOS, option 5 is displayed only if your system has more than one hard disk.

***Step 2.** Choose option 1.*

A menu appears, which looks like this one:

```
            Create DOS Partition or Logical DOS Drive

Current fixed disk drive:1

            1. Create Primary DOS Partition
            2. Create Extended DOS Partition
            3. Create Logical DOS Drive(s) in the
               Extended DOS Partition

Enter choice:[ ]
```

The heading will be slightly different with DOS V3.3, but the menu choices are the same. The primary DOS partition is the bootable partition (under DOS V4), and cannot be divided into more than one logical drive. If you wish to have more than one logical drive, you must divide your disk into both primary and extended DOS partitions. Menu option 3 will let you create those logical drives.

DOS V4 also will let you create a primary DOS partition that is larger than 32 megabytes. The extended DOS partition can also be larger than 32 megabytes, and the logical drives there *also* may be larger than 32 megabytes. Note that you cannot create a partition of exactly 32 megabytes; if you choose 32, FDISK will substitute 31.

Step 3. *Choose option 1 to create the primary DOS partition.*

The screen you will see will look like the one that follows.

```
                Create Primary DOS Partition

  Current fixed disk drive: 1

  Do you wish to use the maximum available size for a Primary
  DOS Partition and make the partition active (Y/N).....? [Y]

  Press Esc to return to FDISK options.
```

Step 4. *Decide how much of your hard disk you want to allocate to the primary partition.*

If you want to use the entire hard disk for a single partition, even if you are using DOS V4 and the partition is more than 32 megabytes, type **Y**. If you enter **N**, DOS will report on available space and ask you to enter the amount to be set aside for this partition. With DOS V3.3, enter the number of megabytes to be allocated. With DOS V4, you may enter either the number of megabytes or the percentage. For example, with a 44-megabyte disk, you could type either **22** or **50%** to allocate half for the primary partition. When you have selected the amount of space to allocate, press Enter. You'll be returned to the main menu.

Step 5. *Choose option 2 from the main menu to make the primary partition active, so that you'll be able to use it to boot your computer.*

If you want to create additional DOS partitions, because you have another physical hard disk or have not set aside the entire disk as the primary partition, follow the next steps. Otherwise, press the Esc key and you are finished with FDISK. Go to the section "Using FORMAT To Prepare Your Hard Disk" to format your hard disk.

If additional partitions can be created on your disk, FDISK will automatically take you to the next menu, which looks like this:

```
        Create Extended DOS Partition

Current fixed disk drive:1

Partition Status  Type   Size in Mbytes  Percentage of Disk Used
C:1       A     PRI DOS      22                   50%

Total disk space is 22 Mbytes (1 Mbyte = 1048576 bytes)
Maximum space available for partition is 22 Mbytes (50%)

Enter partition size in Mbytes or percent of disk space (%) to
create an Extended DOS Partition.....................[22]

Press Esc to return to FDISK options.
```

`C:1` in the first column indicates the drive letter and the number assigned to the DOS partition. The `A` indicates that the partition is active; `PRI DOS` shows that it is the primary DOS partition. The Size and Percentage columns show the amount of space already used for the primary partition and any other DOS partitions you may already have created.

Step 6. *Enter the amount of space to be allocated to the next partition.*

Just as you did when you created your primary partition, you may enter the size of the extended partition(s) in megabytes or (with DOS V4) in percentages.

Step 7. *Now create logical disk drives in the partition if you want to divide it into separate drives.*

Each time you create an extended partition, FDISK will show you the following screen:

```
  Create Logical DOS Drive(s) in the Extended DOS Partition

No logical drives defined

Total Extended DOS Partition size is 22 Mbytes
(1 MByte = 1048857 bytes)
Maximum space available for logical drive is 22 Mbytes (50%)

Enter logical drive size in Mbytes or percent of disk space
(%)...[ ]

Press Esc to return to FDISK options.
```

DOS V3.3 does not show all the information provided in this DOS V4 screen, such as the size of a megabyte, or the percentage of the disk used. However, the concept is the same. Enter the space in megabytes only, rather than percentages, if you have DOS V3.3.

Step 8. *Repeat steps 5–7 to create additional partitions.*

When you have created all the partitions and logical drives you want, press Esc to return to the FDISK menu. You may exit to DOS from there.

Using FORMAT To Prepare Your Hard Disk

The FORMAT command performs several functions. It lays out individual sections of the disk that DOS uses to store your programs and data. FORMAT creates the housekeeping areas so that DOS can keep track of your programs and files. The command also places the key parts of DOS on your hard disk.

To do its job, FORMAT permanently removes any files stored on the hard disk. *If you have on your hard disk any files you want to keep, do not perform the steps in this section.*

Step 1. *With the DOS start-up disk in the floppy disk drive and the door closed, start the FORMAT program.*

You do so by typing

FORMAT C: /S /V

Then press Enter. The light on the floppy disk drive should go on for a few seconds. You should see this message:

```
WARNING, ALL DATA ON NON-REMOVABLE DISK

DRIVE C: WILL BE LOST! Proceed with Format (Y/N)?
```

Step 2. *Because you do want to format the hard disk, press Y and then press Enter.*

You should see the following message:

```
Formatting...
```

The red light for the hard disk will stay on for two to four minutes, depending on the type of hard disk you have.

After FORMAT has completed its first stage, the message `Format complete` will appear. Within a few seconds, FORMAT will provide the message `System transferred`. This informs you that DOS has been placed on the hard disk.

Then you should see the following message:

```
Volume label (11 characters, ENTER for none)?
```

Step 3. *Type a volume name that contains up to 11 characters.*

This name will appear whenever you display a list of files on the hard disk and at several other times also. After you have typed the name, press Enter.

Step 4. *Test the setup of your computer.*

Open the floppy disk drive door and press the Ctrl-Alt-Del keys (hold down the first two keys while pressing the Del key). This sequence should restart DOS from the hard disk.

The floppy disk drive light should come on, followed by the hard disk drive light. Then the computer should beep. The computer will attempt to load DOS from the floppy disk drive. Several seconds later, the computer will try to load DOS from the hard disk. Within 20 seconds, DOS should ask you for the date.

Answer the date and time messages if they appear, as you did in steps 1 and 2 in the first section. You have now successfully prepared the hard disk to start DOS. Go to the next section on setting up files on your hard disk for daily use.

If you don't see a message about the date, either you did not press the Ctrl, Alt, and Del keys correctly, or you were not successful in preparing your hard disk. Try pressing Ctrl-Alt-Del again. If you do not see the date and time messages within 30 seconds, close the disk drive door, restart DOS from the floppy disk (press Ctrl-Alt-Del again), return to step 1, and repeat the instructions in this section again.

Setting Up the Hard Disk for Daily Use

Because only basic setup commands are used for this setup procedure, the procedure will work for DOS V2, V3, and V4. However, some of the special capabilities of DOS V3 and V4 are not used. (If you have DOS V4, you may prefer to set up your hard disk using SELECT.)

Your hard disk is now ready to store and retrieve information under DOS. In this section, you will copy the DOS programs to the hard disk and also create some files for customizing DOS. If your computer uses minifloppy disk drives, you will have extra steps in this section because DOS comes on more than one minifloppy disk.

First, you'll restart DOS to ensure that your computer is working properly. In this section, remember to end each line you are asked to type by pressing Enter.

Step 1. *Check that the DOS start-up disk is in the floppy disk drive and that the disk drive door is closed.*

Step 2. *Make the hard disk the current disk drive.*

You do so by typing **C:**. Don't forget to press the Enter key afterward. The DOS prompt should look like this:

```
C>
```

Step 3. *Create a subdirectory called BIN by typing*

MKDIR \BIN

Later you will place the DOS programs in this subdirectory.

Step 4. *Create a subdirectory in BIN called SYS by typing*

MKDIR \BIN\SYS

Here you will place the DOS device-driver software (files ending in SYS).

Step 5. *Create another subdirectory in BIN called SAMPLES by typing*

MKDIR \BIN\SAMPLES

This subdirectory is where you will place the BASIC programs on the DOS Supplemental Programs or Operating disk.

Step 6. *Make BIN the current directory by typing*

CHDIR \BIN

Step 7. *Copy all files from the DOS start-up disk to the current directory (BIN) on the hard disk by typing*

COPY A:*.* C: /V

You will see the name of each file as it is copied. Then you'll see a message like the following:

```
36 files copied
```

Remember that the number of files on the disk (and the number of files copied) will vary, depending on your version of DOS.

Step 8. *Copy all files ending with SYS to the SYS directory by typing*

COPY *.SYS SYS /V

You will see a message indicating that two files have been copied.

Step 9. *Remove the two SYS files from the BIN subdirectory by typing*

ERASE *.SYS

Because you have safely copied (in step 8) the files to the SYS subdirectory, you can remove them from BIN.

Step 10. *Take the DOS start-up disk out of drive A:. Put the disk back into its protective envelope. Put the DOS Operating disk into drive A: and close the drive door.*

Step 11. *Copy to the current directory (BIN) the program files with .COM extensions on the Operating disk by typing*

COPY A:*.COM C: /V

Step 12. *Copy to the current directory (BIN) the program files with .EXE extensions on the Operating disk by typing*

COPY A:*.EXE C: /V

Step 13. *Copy the BASIC sample programs to the SAMPLES subdirectory of the hard disk by typing*

COPY A:*.BAS \SAMPLES /V

You should see a message indicating that 13 files have been copied.

You have now copied most of the files from the DOS disk to the hard disk. The files you did not copy are not important.

Step 14. *Remove the Operating disk and store it and the start-up disk in a safe place. (Remember that if you have DOS V4, your disks have other names, such as Install and Operating 1, Operating 2, and Operating 3.)*

Step 15. *Make the root (topmost) directory of the hard disk the current directory by typing*

**CD **

Step 16. *Begin to create a CONFIG.SYS file by typing the following command:*

COPY CON CONFIG.SYS

The DOS prompt (`C>`) will disappear.

Step 17. *Now type the following lines. Be sure to check your typing before you press Enter at the end of each line. If you make a mistake and you have already pressed Enter, perform step 20 and then return to step 16.*

BUFFERS = 20
DEVICE = c:\bin\sys\ansi.sys
FILES = 20
FCBS = 12,12 LASTDRIVE = F

Step 18. *If you live in the United States or the English-speaking provinces of Canada, type the following line:*

COUNTRY = 001

If you live somewhere else, see Chapter 15 for the number you should use in place of 001. Then type the following line, replacing the **xxx** with the correct three-digit code:

COUNTRY = xxx

Step 19. *If you want to use a RAM disk on a PC XT or an AT with 640K of memory or less, type the following line:*

DEVICE = c:\bin\sys\vdisk.sys 180 512 64

If you want to use a RAM disk on a Personal Computer AT with more than 640K of memory, type the following line instead:

DEVICE = c:\bin\sys\vdisk.sys 180 512 64 /e:8

If you do not want to use a RAM disk, skip this step and continue with step 20.

Step 20. *The cursor should be at the beginning of a blank line. If not, press Enter. Then press the F6 special-function key.*

A `^Z` should appear. Press Enter again. The hard disk light should come on briefly, and you should see the message `1 file(s) copied`. The DOS prompt will reappear.

You have just created a CONFIG.SYS file for your computer.

Step 21. *To begin creating an AUTOEXEC.BAT file, type the following line:*

COPY CON AUTOEXEC.BAT

The DOS prompt should disappear again.

Step 22. *Now type the following lines:*

ECHO OFF
PATH=C:\BIN;C:

As before, check your typing before you press Enter at the end of each line (for this step and the next two steps). If you notice a mistake after you have pressed Enter, perform step 26 and then return to step 21.

Step 23. *If you are in the United States or the English-speaking provinces of Canada, go to step 24. Otherwise, see Chapter 15 for the number to use for the keyboard code. Type the following line, replacing the* **xx** *with the correct two-letter code:*

KEYBxx

Step 24. *If you have an IBM Color Graphics Adapter (CGA) on your computer, type the following line as well:*

GRAFTABL

Step 25. *If you have a Personal Computer AT, continue with step 26. Otherwise, type the following lines:*

DATE
TIME

Step 26. *The cursor should be at the beginning of a blank line. If not, press Enter. Then press the F6 special-function key.*

A `^Z` should appear. Press Enter again. The hard disk light should come on briefly, and you should see the message `1 file(s) copied`. The DOS prompt (`C>`) will reappear.

You have just created an AUTOEXEC.BAT file for your computer.

Step 27. *Test the setup you have just created. Make sure that the floppy disk drive door is open. Press the Ctrl-Alt-Del sequence to restart DOS. In one minute, you should see the following lines if you used the VDISK line in your CONFIG.SYS file:*

```
VDISK  Version 2.0 virtual disk D:
VDISK Buffer size:          180 KB
VDISK Sector size:          512
VDISK Directory entries:     64
VDISK Transfer size:          8
```

If you are not using a Personal Computer AT or PS/2, DOS will ask for the date and time. Provide this information. The DOS prompt, `C>`, should appear. If you are using a Personal Computer AT, the DOS prompt should appear; you should not see a request for the date or time.

If you see a message about a file not found, a syntax error, or some other problem, repeat steps 16–27. If the same message appears again, repeat the entire procedure, beginning at the point where you create the \BIN directory.

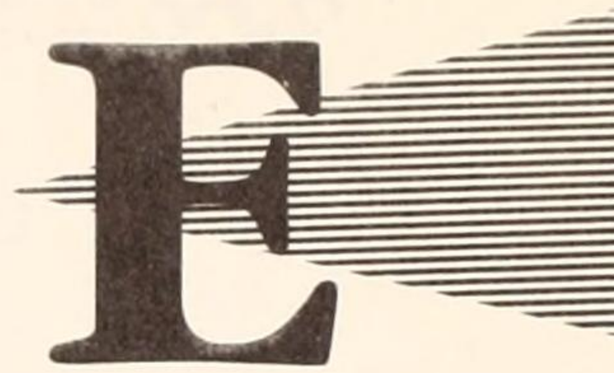
The final test is to run the CHKDSK program. Type **CHKDSK**. If you see a message that says something about a bad command or a file not found, return to step 21 and try creating your AUTOEXEC.BAT file again. If, after restarting DOS and typing CHKDSK, you get the same message, the DOS files did not get copied correctly. Try repeating steps 6, 7, and 9.

You're done! DOS is set up on your computer and is ready to go. When you start your computer, make sure that the door to the floppy disk drive is open. When you turn on the power or press Ctrl-Alt-Del, DOS will automatically start from the hard disk.

A Note to Personal Computer AT Owners

If you discovered that the date or time for your computer was incorrect, the correct date or time you gave to DOS is kept until the computer is turned off or until DOS is restarted. When you use DOS's DATE and TIME commands to set the computer's date and time, DOS versions prior to V3.3 do not set the battery-backed clock/calendar of the AT.

To change the internal date and time of the Personal Computer AT, you must run the SETUP program again. Directions for using SETUP are in the Installation and Setup manual that comes with your computer.



ANSI Terminal Codes

All ANSI terminal codes are preceded by an Escape (ESC) character and a left bracket ([). The Escape character is 27 decimal or 1b hexadecimal.

Cursor-Control Sequences

For the cursor-control sequences, if a value is omitted, the default value of 1 is used.

Cursor Position

Horizontal and Vertical Position

ESC[#;#H
ESC[#;#f

The first # is the row (vertical coordinate); the second # is the column (horizontal coordinate). The starting value for either coordinate is 1 (also the default value).

Cursor Up

ESC[#A

is the number of rows to move up.

Cursor Down

ESC[#B

is the number of rows to move down.

Cursor Forward

ESC[#C

is the number of columns to move forward (right).

Cursor Backward

ESC[#D # is the number of columns to move backward (left).

Device Status Report

ESC[6n "Inputs" through the keyboard a cursor report on the current cursor position.

Cursor Position Report

ESC[#;#R Reports the current cursor position. The string is returned by the ANSI console and is eight characters long. The first # is the two-digit row number; the second # is the two-digit column number.

Save Cursor Position

ESC[s "Saves" the current cursor position in the ANSI.SYS driver. The position can be restored by using ESC[u.

Restore Cursor Position

ESC[u Sets the cursor to the horizontal and vertical position that was "saved."

Erasing

Erase Display

ESC[2J Erases the display and also moves the cursor to the Home position (the upper left corner of the screen).

Erase to End of Line

ESC[K Erases from the current cursor position to the end of the current line.

Modes of Operation

For all ANSI-operation-mode control codes, if a parameter is omitted, the default value of 0 is used.

Set Graphics Rendition

ESC[#;...;#m Sets the character attributes by the following parameters. The attributes remain in effect until the next graphics rendition command.

Parameter	*Meaning*
0	All attributes off (normally white on black)
1	Bold on (high intensity)
4	Underscore on (IBM Monochrome Display only)
5	Blink on
7	Reverse (inverse) video on
8	Cancel on (invisible characters black on black)
30	Black foreground
31	Red foreground
32	Green foreground
33	Yellow foreground
34	Blue foreground
35	Magenta foreground
36	Cyan foreground
37	White foreground
40	Black background
41	Red background
42	Green background
43	Yellow background
44	Blue background
45	Magenta background
46	Cyan background
47	White background

Set Screen Mode

Esc[=#h Sets the screen width or type based on #.

for parameter 7 only, the syntax is

Esc[=7h
Esc[?7 Turns on the wrap at end-of-line; long lines automatically wrap onto subsequent lines.

Reset Screen Mode

Esc[=#1 Resets the screen width or type based on # (in the following list).

for parameter 7 only, the syntax is

Esc[?71
Esc[=71 Turns off wrap at end-of-line; long lines are truncated at the end-of-line.

Parameter	*Meaning*
0	40 × 25 monochrome
1	40 × 25 color
2	80 × 25 monochrome
3	80 × 25 color
4	320 × 200 color
5	320 × 200 monochrome
6	640 × 200 monochrome
7	Wrap at end-of-line (set mode), or do not wrap and discard characters past end of line (reset mode)
*14	640 × 200 color
*15	640 × 350 monochrome
*16	640 × 350 color
*17	640 × 480 monochrome
*18	640 × 480 color
*19	320 × 200 color

*Available with DOS V4.0 and later versions.

Keyboard Key Reassignment

Reassign Keyboard Keys

Esc[sequencep Sets the sequence typed by the named key.

sequence can be in the form of #;#;...# (where # represents the one-, two-, or three-digit code for an ASCII character), or *"string"* (a string of characters within double quotation marks), or any combination of the two provided that # and *"string"* are separated by a semicolon.

Esc[#;#;...;p

or

Esc["string"p

or

Esc[#;"string";#;#;"string";#p

The first ASCII code (the first #) defines which key or keystrokes (such as a Ctrl-character combination) are being reassigned. However, if the first code in the sequence is 0 (ASCII Nul), the first and second codes designate an Extended ASCII key sequence. See Appendix B for the set of ASCII codes and the set of Extended ASCII codes.

The remaining numbers (#) or characters within the "string" are the replacement characters typed when that key or keystroke combination is pressed. Any nonnumeric characters used in the replacement must be placed within double quotation marks.

Enable/Disable Extended Keys on Enhanced Keyboards

Esc[1q Enables assignment of the extended keys on the Enhanced Keyboard. The sequence is the same as using the /X switch for the ANSI.SYS directive.

Esc[Oq Disables assignment of the extended keys on the Enhanced Keyboard.

ANSI.SYS Keyboard Scan Codes

Table 12.5
Scan Codes for IBM PC

Key	*Unshift*	*Shift*	*Control*	*Alt*
A	97	65	1	0;30
B	98	66	2	0;48
C	99	67	3	0;46
D	100	68	4	0;32
E	101	69	5	0;18
F	102	70	6	0;33
G	103	71	7	0;34
H	104	72	8	0;35
I	105	73	9	0;23
J	106	74	10	0;36
K	107	75	11	0;37
L	108	76	12	0;38
M	109	77	13	0;50
N	110	78	14	0;49
O	111	79	15	0;24
P	112	80	16	0;25
Q	113	81	17	0;16
R	114	82	18	0;19
S	115	83	19	0;31
T	116	84	20	0;20
U	117	85	21	0;22
V	118	86	22	0;47
W	119	87	23	0;17
X	120	88	24	0;45
Y	121	89	25	0;21
Z	122	90	26	0;44
1	49	33		0;120
2	50	64		0;121
3	51	35		0;122
4	52	36		0;123
5	53	37		0;124
6	54	94		0;125
7	55	38		0;126
8	56	42		0;127
9	57	40		0;128
0	48	41		0;129
–	45	95		0;130
=	61	43		0;131
Tab	9	0;15		
Space	57			
Null	0;3			

Table 12.6
Extended Scan Codes for Numeric Keypad and Function Keys

Key	*Unshift*	*Shift*	*Control*	*Alt*
F1	0;59	0;84	0;94	0;104
F2	0;60	0;85	0;95	0;105
F3	0;61	0;86	0;96	0;106
F4	0;62	0;87	0;97	0;107
F5	0;63	0;88	0;98	0;108
F6	0;64	0;89	0;99	0;109
F7	0;65	0;90	0;100	0;110
F8	0;66	0;91	0;101	0;111
F9	0;67	0;92	0;102	0;112
F10	0;68	0;93	0;103	0;113
F11	0;133	0;135	0;137	0;139
F12	0;134	0;136	0;138	0;140
Home	0;71	55	0;119	
Crs-Up	0;72	56		
PgUp	0;73	57	0;132	
Crs-Lf	0;75	52	0;115	
Crs-Dn	0;77	54	0;116	
End	0;79	49	0;117	
PgDn	0;81	51	0;118	
Ins	0;82	48		
Del	0;83	46		
PrtSc			0;114	

Other special code combinations include the following:

Key Combination	*Code*
Alt-hyphen (–)	130
Alt-equal sign (=)	131
Ctrl-asterisk (* on numeric keypad)	114
Shift-Tab	15
Ctrl-Alt-Num Lock	55
Ctrl-Alt-Scroll Lock	56

Code Page Tables

Code page 437 (United States)

Hex Digits 1st → 2nd ↓	0-	1-	2-	3-	4-	5-	6-	7-		8-	9-	A-	B-	C-	D-	E-	F-
-0		►		0	@	P	`	p		Ç	É	á	▒	└	╨	α	≡
-1	☺	◄	!	1	A	Q	a	q		ü	æ	í	▒	┴	╤	β	±
-2	☻	↕	"	2	B	R	b	r		é	Æ	ó	▓	┬	╥	Γ	≥
-3	♥	!!	#	3	C	S	c	s		â	ô	ú	│	├	╙	π	≤
-4	♦	¶	$	4	D	T	d	t		ä	ö	ñ	┤	─	╘	Σ	⌠
-5	♣	§	%	5	E	U	e	u		à	ò	Ñ	╡	┼	╒	σ	⌡
-6	♠	▬	&	6	F	V	f	v		å	û	ª	╢	╞	╓	µ	÷
-7	•	↨	'	7	G	W	g	w		ç	ù	º	╖	╟	╫	τ	≈
-8	◘	↑	(	8	H	X	h	x		ê	ÿ	¿	╕	╚	╪	Φ	°
-9	○	↓	)	9	I	Y	i	y		ë	Ö	⌐	╣	╔	┘	Θ	∙
-A	■	→	*	:	J	Z	j	z		è	Ü	¬	║	╩	┌	Ω	·
-B	♂	←	+	;	K	[	k	{		ï	¢	½	╗	╦	▓	δ	√
-C	♀	∟	,	<	L	\	l	|		î	£	¼	╝	╠	▬	∞	n
-D	♪	↔	-	=	M	]	m	}		ì	¥	¡	╜	═	▌	ϕ	2
-E	♫	▲	.	>	N	^	n	~		Ä	Pt	«	╛	╬	▐	ε	■
-F	☼	▼	/	?	O	_	o	⌂		Å	ƒ	»	┐	╧	▀	∩	

Code page 860 (Portugal)

Hex Digits 1st → 2nd ↓	0-	1-	2-	3-	4-	5-	6-	7-		8-	9-	A-	B-	C-	D-	E-	F-
-0		►		0	@	P	`	p		Ç	É	á	▒	└	╨	α	≡
-1	☺	◄	!	1	A	Q	a	q		ü	À	í	▒	┴	╤	β	±
-2	☻	↕	"	2	B	R	b	r		é	È	ó	▒	┬	╥	Γ	≥
-3	♥	!!	#	3	C	S	c	s		â	ô	ú	│	├	╙	π	≤
-4	♦	¶	$	4	D	T	d	t		ã	õ	ñ	┤	─	╘	Σ	⌠
-5	♣	§	%	5	E	U	e	u		à	ò	Ñ	╡	┼	╒	σ	⌡
-6	♠	▬	&	6	F	V	f	v		Á	Ú	ª	╢	╞	╓	µ	÷
-7	•	↨	'	7	G	W	g	w		ç	ù	º	╖	╟	╫	τ	≈
-8	◘	↑	(	8	H	X	h	x		ê	Ì	¿	╕	╚	╪	Φ	°
-9	○	↓	)	9	I	Y	i	y		Ê	Õ	Ò	╣	╔	┘	Θ	•
-A	■	→	*	:	J	Z	j	z		è	Ü	¬	║	╩	┌	Ω	·
-B	♂	←	+	;	K	[	k	{		Ï	¢	½	╗	╦	■	δ	√
-C	♀	∟	,	<	L	\	l	|		Ô	£	¼	╝	╠	▬	∞	ⁿ
-D	♪	↔	-	=	M	]	m	}		ì	Ù	¡	╜	═	▌	ϕ	²
-E	♫	▲	.	>	N	^	n	~		Ã	Pt	«	╛	╬	▐	ε	■
-F	☼	▼	/	?	O	_	o	⌂		Â	Ó	»	┐	╧	▀	∩	

Code page 850 (Multilingual)

Hex Digits 1st → 2nd ↓	0-	1-	2-	3-	4-	5-	6-	7-		8-	9-	A-	B-	C-	D-	E-	F-
-0		►		0	@	P	`	p		Ç	É	á	▒	└	ð	Ó	-
-1	☺	◄	!	1	A	Q	a	q		ü	æ	í	▒	┴	Đ	β	±
-2	☻	↕	"	2	B	R	b	r		é	Æ	ó	▒	┬	Ê	Ô	=
-3	♥	!!	#	3	C	S	c	s		â	ô	ú	│	├	Ë	Ò	¾
-4	♦	¶	$	4	D	T	d	t		ä	ö	ñ	┤	─	È	õ	¶
-5	♣	§	%	5	E	U	e	u		à	ò	Ñ	Á	┼	ı	Õ	§
-6	♠	▬	&	6	F	V	f	v		å	û	ª	Â	ã	Í	µ	÷
-7	•	↨	'	7	G	W	g	w		ç	ù	º	À	Ã	Î	þ	¸
-8	◘	↑	(	8	H	X	h	x		ê	ÿ	¿	©	╚	Ï	Þ	°
-9	○	↓	)	9	I	Y	i	y		ë	Ö	®	╣	╔	┘	Ú	¨
-A	■	→	*	:	J	Z	j	z		è	Ü	¬	║	╩	┌	Û	·
-B	♂	←	+	;	K	[	k	{		ï	ø	½	╗	╦	■	Ù	¹
-C	♀	∟	,	<	L	\	l	|		î	£	¼	╝	╠	▬	ý	³
-D	♪	↔	-	=	M	]	m	}		ì	Ø	¡	¢	═	¦	Ý	²
-E	♫	▲	.	>	N	^	n	~		Ä	×	«	¥	╬	ì	¯	■
-F	☼	▼	/	?	O	_	o	⌂		Å	ƒ	»	┐	¤	▬	´	

Code page 863 (Canada-French)

Hex Digits 1st → 2nd ↓	0-	1-	2-	3-	4-	5-	6-	7-		8-	9-	A-	B-	C-	D-	E-	F-
-0		►		0	@	P	`	p		Ç	É	¦	▒	└	╨	α	≡
-1	☺	◄	!	1	A	Q	a	q		ü	È	´	▒	┴	╤	β	±
-2	●	↕	"	2	B	R	b	r		é	Ê	ó	▒	┬	╥	Γ	≥
-3	♥	!!	#	3	C	S	c	s		â	ô	ú	│	├	╙	π	≤
-4	♦	¶	$	4	D	T	d	t		Â	Ë	¨	┤	─	╘	Σ	⌠
-5	♣	§	%	5	E	U	e	u		à	Ï	˜	╡	┼	╒	σ	⌡
-6	♠	▬	&	6	F	V	f	v		¶	û	³	╢	╞	╓	µ	÷
-7	•	↨	'	7	G	W	g	w		ç	ú	˙	╖	╟	╫	τ	≈
-8	◘	↑	(	8	H	X	h	x		ê	¤	Î	╕	╚	╪	Φ	°
-9	○	↓	)	9	I	Y	i	y		ë	Ô	⌐	╣	╔	┘	Θ	•
-A	◙	→	*	:	J	Z	j	z		è	Ü	¬	║	╩	┌	Ω	·
-B	♂	←	+	;	K	[	k	{		ï	¢	½	╗	╦	█	δ	√
-C	♀	∟	,	<	L	\	l	|		î	£	¼	╝	╠	▄	∞	ⁿ
-D	♪	↔	-	=	M	]	m	}		=	Ù	¾	╜	═	▌	ϕ	²
-E	♫	▲	.	>	N	^	n	~		À	Û	«	╛	╬	▐	ε	■
-F	☼	▼	/	?	O	_	o	⌂		§	ƒ	»	┐	╧	▀	∩	

Code page 865 (Norway)

Hex Digits 1st → 2nd ↓	0-	1-	2-	3-	4-	5-	6-	7-		8-	9-	A-	B-	C-	D-	E-	F-
-0		►		0	@	P	`	p		Ç	É	á	▒	└	╨	α	≡
-1	☺	◄	!	1	A	Q	a	q		ü	æ	í	▒	┴	╤	β	±
-2	☻	↕	"	2	B	R	b	r		é	Æ	ó	▓	┬	╥	Γ	≥
-3	♥	!!	#	3	C	S	c	s		â	ô	ú	│	├	╙	π	≤
-4	♦	¶	$	4	D	T	d	t		ä	ö	ñ	┤	─	╘	Σ	⌠
-5	♣	§	%	5	E	U	e	u		à	ò	Ñ	╡	┼	╒	σ	⌡
-6	♠	▬	&	6	F	V	f	v		å	û	ª	╢	╞	╓	µ	÷
-7	•	↨	'	7	G	W	g	w		ç	ú	º	╖	╟	╫	τ	≈
-8	◘	↑	(	8	H	X	h	x		ê	ÿ	¿	╕	╚	╪	Φ	°
-9	○	↓	)	9	I	Y	i	y		ë	Ö	⌐	╣	╔	┘	Θ	•
-A	◙	→	*	:	J	Z	j	z		è	Ü	¬	║	╩	┌	Ω	·
-B	♂	←	+	;	K	[	k	{		ï	ø	½	╗	╦	■	δ	√
-C	♀	∟	,	<	L	\	l	|		î	£	¼	╝	╠	▬	∞	n
-D	♪	↔	-	=	M	]	m	}		ì	Ø	¡	╜	═	▌	ϕ	2
-E	♫	▲	.	>	N	^	n	~		Ä	Pt	«	╛	╬	▐	ε	■
-F	☼	▼	/	?	O	_	o	⌂		Å	ƒ	¤	┐	╧	▀	∩	

The tables in this Appendix are reproduced courtesy of IBM.

Index

$$$ extension, 375, 381-382
/1 switch
  DISKCOMP command, 158, 660-661
  DISKCOPY command, 149, 664-665
  FORMAT command, 681, 683
/3 switch
  FORMAT command, 127
/4 switch
  FORMAT command, 127, 681, 683, 792
/8 switch
  DISKCOMP command, 158, 660-661
  FORMAT command, 127, 681
30-line color (DOS Shell) display mode, 530
386-based computer, 304, 311-312
8086 microprocessor, 13, 71
8088 microprocessor, 13, 71
80286 microprocessor, 13
80386 microprocessor, 13
80486 microprocessor, 13
@ key, 403, 411
@ subcommand, 432

A

+A/-A switch
  ATTRIB command, 794
/A switch
  BACKUP command, 291-292, 294-296, 607, 609, 611, 613-615
  COPY command, 273-277, 653-655
  FC command, 269, 676
  REPLACE command, 161-164, 745-747
  RESTORE command, 298
  SMARTDRV.SYS device driver, 468
  TREE command, 209, 777
  XCOPY command, 215, 217-219, 606, 782, 784
/A:date switch
  RESTORE command, 752-753
/ASC (DOS Shell) option, 573
/AUTOTEST switch
  FORMAT command, 684
A (append lines) command, EDLIN program, 384, 386
Abort edit (Y/N)? message, 393
Access denied filename message, 246, 606, 671, 735, 748-749, 786
Active code page for device devicename is codepage message, 720
Active code page not available from CON device message, 698
adapter
  disk controller, 100
  display, 99
Add filename? (Y/N) Adding filename message, 749
Add Group (DOS Shell) panel, 535, 548
Add Program (DOS Shell) panel, 535, 538-542
Add... (DOS Shell) option, 535, 537-538, 548, 550-551
address bus, 74
aliases
  drive name for a path, 348-350
  files directory, 189, 191, 194, 198-199, 223
All files canceled by operator message, 735
All specified file(s) are contiguous message, 633
Allocation error, size adjusted message, 168
allocation size error repair with CHKDSK, 170-172
alphanumeric keys with Alt key, ANSI.SYS keyboard scan codes, 834
Alt key, 90
Alt-F1 keys, 815
ANSI terminal codes, 829, 831-833
  cursor-control sequences, 829-830
  keyboard key reassignment, 833
  modes of operation, 831-832
ANSI.SYS device driver, 5-6, 105-106, 319-321, 325-326, 383, 417-430, 450-451, 482, 646, 797
  communicating with, 419-422
  cursor movement, 421-422
  enhancing menus, 450-451
  keyboard
    commands, 425-429
    scan codes, 834-835
      alphanumeric keys with Alt key, 834
      cursor pad keys, 835
      function keys, 834
  parameters
    /K, 797
    /L, 797
    /X, 797
  screen commands, 423-424
    color monitor, 424
    monochrome monitor, 423
APPEND already installed message, 602
APPEND command, 79, 81, 182, 199-201, 205, 229, 327, 344, 347-348, 598-602, 730, 753, 783, 786, 795, 797
  environment switch, 200-203, 206
  extended search switch, 200-204, 206-208

switches
/E, 200-203, 206, 599-601
/PATH ON/OFF, 201, 203-204, 206, 599
/X ON/OFF, 200-204, 206-208, 599-602, 783, 785,
syntax, 201-204
append lines with EDLIN, 384, 386
APPEND/ASSIGN Conflict message, 602
APPEND/TopView Conflict message, 602
directory defaults, 227
files, write-protect, 246
installation, 227
ARC file extension, 236
archiving
attribute, 243, 247, 249-250
disks, 290
files, 245, 247, 249-250
Arrange (DOS Shell) menu, 563-564
arrow keys, scrolling text in DOS Shell, 531
ASC file extension, 236
ASCII
binary values, 368-370
bytes, 239-241
characters, 233, 383
codes, 5, 233, 383, 801-806
extended, 806-811
control characters, 143
end-of-file (EOF) character, 239-241, 261, 272, 302, 381
files, 174, 177, 239-241, 272-276, 281, 337, 364, 366-370, 373-374, 381, 400, 418, 456, 617, 653
comparisons, FC command, 266, 268-270
RECOVER, 174
form feed character, 261
mode, 555
sorting, 766-769
values, 233
ASK.COM program, 433
ASM file extension, 236
Assembler language, 25
program, 74-75
ASSIGN command, 79, 83, 133, 144, 159, 206, 221, 333, 346, 370, 600-604, 608, 614, 630, 632-633, 659-660, 665, 683, 685, 692, 700, 706, 741, 753, 770-772
syntax, 344-345
Associate... (DOS Shell) option, 556-557
Asynchronous Communications Adapter protocol, 710-711
Attempted write-protect violation message, 685
ATTRIB command, 79, 81, 217, 231, 243-251, 294, 302, 586, 605-607, 671, 749, 785-787, 792, 794-795
switches, 242-243
+A/-A, 242-243, 794
/S, 605, 795
syntax, 242-243
attributes
archive, 243, 247, 249-250
field directory entry, 183
read-write, 243-244, 246-247, 249-251
AUTOEXEC.BAT file, 24, 76, 108, 122, 156, 165, 194, 196, 224-226, 260, 284-285, 305, 318, 327, 330, 334, 338, 446-449, 490, 493, 497-498, 519-521, 617, 627, 657, 740, 758-761, 796, 827
international
directives, 523-524
features, 507-509
AUX (device name), 103, 235, 340, 353

B

/B (DOS Shell) option, 573
/B switch
COPY command, 273, 276-277, 285-286, 653-655
EDLIN program, 381
FC command, 268, 270, 676-677
FORMAT command, 147, 681, 683
GRAPHICS command, 335-336, 691
RESTORE command, 297, 796
/B:bufsiz switch
PRINT command, 339-343, 731
/B:date switch
RESTORE command, 752-753
/BACKUP switch
FORMAT command, 684
backslash (\) character, 86, 184, 186, 192, 234
Backspace key, 89, 92
backing up disks, 289-296
restoring files, 751-757
BACKUP command, 33, 79, 81, 139, 172, 176, 218, 227, 231, 244-245, 287-296, 299, 302, 344, 347-348, 433, 565, 604, 607-616, 623, 634, 658, 682, 693, 752-753, 755, 770, 772, 784-785, 792, 795, 797
switches
/A, 291-292, 294-296, 607, 609, 611, 613-615
/D, 291
/D:date switch, 607, 614
/F, 291-293, 295, 608-609, 612-613, 682, 795
/F:size, 291, 608-609
/L, 291, 608, 610, 613, 615, 795
/M, 291, 294, 296, 606-607, 611, 785
/S, 291, 296, 607, 610-611
/T, 291, 795
/T:time switch, 608, 614
syntax, 290-291
Backup Fixed Disk (DOS Shell) option, 565
BACKUP.nnn file, 612
BACKUPID.@@@ file, 612, 755
Bad command message, 596
Bad command or filename message, 199, 582, 779
Bad format call message, 596

Bad or missing Command Interpreter message, 582
Bad or missing filename message, 582-583
Bad or missing Keyboard Definition File message, 698
Bad parameter message, 661
Bad Partition Table message, 685
Bad unit message, 596
BAK file extension, 236, 382, 641
BAS file extension, 236, 556
Basic Input/Output System (BIOS), 20-23, 25, 32, 53, 57, 70, 73, 99, 128
  device driver, 286
  extensions, 76, 107
  layer, 73
BASIC
  language, 25, 75, 234
  program, 353, 556
BASICA program, 225
BAT file extension, 77-78, 236, 194, 200-201, 206, 401, 414, 554, 556, 616-617, 729
Batch command, 616-618
Batch file missing message, 583
batch files, 4, 77-78, 80, 83-84, 399-415, 616-618, 795
  advanced techniques, 5, 431-451
  calling
    at end of batch file, 442-445
    with COMMAND subcommand /C, 445-446
constructing with parameters, 408-410
display message within, 624
  DOSSHELL, 319
formatting diskettes, 402-404
jumping to another part of, 411-414, 621-622
labels, 410-412
missing, 587
naming, 401
notation, 580-581
parameters, 405-406
remarks in, 410-411
rules, 414-415
running from other batch files, 441-446, 618-619
shift line parameters left, 624
subcommands, 431-446
suspend processing, 623
  variable markers, 406-408
batch subcommands
  CALL, 618-619
  display messages while executing, 619-620
  ECHO, 618-620, 623-624
  FOR, 618
  FOR..IN..DO, 620, 624
  GOTO, 621-622
  IF, 622-623
  PAUSE, 618, 623
  REM, 620, 624
  SHIFT, 624
BATCH1.BAT file, 443
BATCH2.BAT file, 443
BATCH3.BAT file, 443
baud rate, 358
BIN file extension, 236, 269
  from EXE files, 672-673
binary
  copy files, 272-276, 285-286
  data, 22
  digit, 42-43, 73, 239-241
  files, 381
    comparisons, FC command, 266, 268-270
    RECOVER, 174
  language layer, 74-75
  level instructions, 71-72
  storage, 42
  values, 73
    ASCII, 368-370
bit (binary digit), 42-43, 73, 239-241
black-and-white graphics (DOS Shell) display mode, 530
block devices
  defining or changing parameters, 647
  driver, 101-102
boldface characters, 7
booting, 3, 75-76
  cold, 107, 196
  disk, 16-17, 107-108, 225
  operating system, 16-17
  process, 69, 72, 76
  record, 128, 131
  warm, 91, 107, 196
BREAK configuration subcommand, 78, 84, 456-457, 625, 643
Break key, 90
BRIDGE.DRV file, 106
buffers
  disk, 458-461
  expanded memory, 461
  floppy disk system, 460
  flushing, 459
  hard disk system, 460
  limiting memory in DOS Shell, 573
  look-ahead, 461
  setting number in memory, 644-645
  size (PRINT), 339-340
BUFFERS configuration subcommand, 84, 456, 458, 491, 644-645, 795, 797
  DOS version 4, 461
  switches
    /X, 461, 644, 797
bus, 74, 98
bytes, 43, 48-50, 58, 71, 73, 128, 239-242
  offset, 130

C

/+c switch
  SORT command, 766
/C string switch
  COMMAND command, 327-328

/C switch
COMMAND subcommand, 445-446, 619, 639
DRIVER.SYS device driver, 476
DRIVPARM configuration subcommand, 647
FC command, 269, 676-677
FIND command, 366, 680-681
PRINT command, 340-343, 732
/CLR (DOS Shell) option, 571-572
/CO1 (DOS Shell) option, 573-574
/CO2 (DOS Shell) option, 573-574
/CO3 (DOS Shell) option, 573-574
/COLOR (DOS Shell) option, 570
/COM2 (DOS Shell) option, 574
C (C source file) extension, 236
C (copy lines) EDLIN command, 384, 387-389, 397
C language, 25, 75
caches *see* disk caches
CALL batch subcommand, 78, 83, 415, 432, 442-445, 618-619, 795
DOS version 3.3 and 4.0, 442
`Cannot CHDIR to directory Tree past this point not processed` message, 633
`Cannot CHDIR to root Processing cannot continue` message, 633
`Cannot CHKDSK a Network drive` message, 633
`Cannot CHKDSK a SUBSTed or ASSIGNed drive` message, 633
`Cannot DISKCOMP to or from network drive` message, 661
`Cannot DISKCOPY to or from a network drive` message, 667
`Cannot do binary reads from a device` message, 655
`Cannot execute A: program_name` message, 761
`Cannot execute FORMAT` message, 612
`Cannot format an ASSIGNed or SUBSTituted drive` message, 685
`Cannot FORMAT nonremovable drive d:` message, 612-613
`Cannot JOIN a network drive` message, 695
`Cannot LABEL a network drive` message, 702
`Cannot load COMMAND, system halted` message, 583
`Cannot perform a cyclic copy` message, 786
`Cannot RECOVER a Network drive` message, 742
`Cannot start COMMAND, exiting` message, 583-584
`Cannot SUBST a network drive` message, 772
`Cannot SYS to a Network drive` message, 774
`Cannot use FASTOPEN for drive d:` message, 674
`Cannot use PRINT - Use NET PRINT` message, 735
`Cannot XCOPY to a reserved device` message, 786
`Cannot XCOPY from a reserved device` message, 786
Caps Lock key, 90-91
CBL file extension, 236
CD command, 404, 627-629, 826
CD ROM file storage, 31
CD.ASC file, 404, 405
central processing unit (CPU), 21-22, 71-72, 74-75, 94
expanded memory, 111-113
extended memory, 111
function, 73
CFG file extension, 236
CGA (Color Graphics Adapter), 22, 99, 324, 326, 529
additional character sets, 688-690
Change Colors (DOS Shell) utility, 564, 570
Change Disk Cache Settings screen, 466
Change Group (DOS Shell) panel, 548
Change Program (DOS Shell) panel, 546
Change... (DOS Shell) option, 537, 546, 548
characters
backslash (\), 234
character set (CHCP), changing, 321-322
device driver, 100-104
international, 480-484, 645, 727-729, 759
meta-string, 738-739
repeating rate, 718-719
string, EDLIN, 393-394
CHCP command, 78, 82, 85, 303-304, 321, 332, 498, 507-509, 516-517, 524, 625-627, 715, 718, 722, 727-728, 795
syntax, 322
CHDIR (CD) command, 59, 78, 81, 181, 192-194, 223, 225, 229, 344, 347-348, 627-629, 659, 825
syntax, 191
`CHDIR.. failed trying alternate method` message, 634
CHECKIT.BAT file, 436-437
chip, 71
CHKDSK command, 30, 79-80, 122, 129, 139, 141, 145, 165, 167-169, 172, 176, 179, 293, 299-300, 334, 492, 585-586, 588, 593-594, 597, 604, 629-638, 663, 666, 669, 693-695, 701, 755, 769-771, 777-778, 797, 828
directory entries, 189
repair allocation size error, 170-172
switches
/F, 166, 586, 629-631, 634-638
/V, 166, 213, 629, 631, 637, 659
syntax, 165-166, 213
CHKDSK.COM file, 413
CHP file extension, 236
clearing screen (CLS), 334, 411
clock
number of ticks for background print (PRINT), 339-340
system, 74
CLOCKS (device name), 103
CLS subcommand, 78, 83, 333, 370, 411, 432, 638-639
syntax, 334
cluster, 48-58, 60-61, 128-129, 166-168
ERASE command, 257
size, 49
starting cluster number directory, 183
valid starting cluster, 189
CNF file extension, 236
COBOL language, 75

Code page nnn not prepared for all devices message, 626
Code page nnn not prepared for system message, 626
Code page requested (codepage) is not valid for given keyboard code message, 698
Code page specified has not been prepared message, 698-699
Code page specified is inconsistent with the selected code page message, 698
code pages, 321-322, 332, 498-504, 645, 689
  activating, 516-517
  changing, 321-322, 508-509, 512-518, 625-627
  checking, 517-518
  defining, 513-516
  displaying multiple, 502-504
  selecting, 712-715
  tables, 837-841
  used with device, 714-715
Code pages cannot be prepared message, 721
Codepage not prepared message, 720
Codepage operation not supported on this device message, 720-721
codes
  ASCII, 5, 233, 239-242, 383, 801-806
  binary, 75
  country, 501
  device control, 240-242
  exit, 433
  machine, 71, 74-75
  scan, 425-429
cold boot, 196
colon (:) key, 86, 410-412
color
  color graphics display, 324
    DOS Shell, 529, 564
Color Graphics Adapter (CGA) display adapter, 324, 326, 499, 519
color monitors, ANSI.SYS screen commands, 424
COM file extension, 77-78, 194, 200-201, 206, 236, 269, 401, 414, 554, 556, 617, 677, 729, 734, 825
COM files from EXE files, 672-673
COM1 (device name), 103, 340, 353-354, 358
COM2 (device name), 340, 353, 358
COM3 (device name), 340, 353, 358
COM4 (device name), 340, 353, 358
comma (,) key, 387, 389-394
COMMAND command, 79, 82, 304, 329, 332, 415, 432, 442, 445-446, 639
  DOS version 3.0 to 3.2, 442
  switches
    /C string, 327-328
    /C, 445-446, 619, 639
    /E:eeeee, 327-328
    /E:size, 639
    /MSG, 327
    /P, 327, 639
  syntax, 327-329
command interpreter, 85-95, 326-327
command line, 85-87
  syntax, 185-186
command parameter, 85-87
command processor, 21, 69-95, 327
  changing default, 651
  current copy, 328-329
  secondary, 328-329
    return to primary, 673
Command Prompt (DOS Shell) option, 529, 552
Command Reference, 579-789
COMMAND.COM file, 17, 21, 23-25, 69-70, 72-73, 76-81, 85-89, 94-98, 107, 116, 122, 127, 132, 145, 148, 165, 189, 224-225, 234-235, 255-256, 275, 292-293, 326, 328, 330, 445, 486-487, 582-585, 587, 589, 592, 595, 601, 651, 688, 754, 756, 763, 773-775
  invoking second copy, 639
  moving to another directory, 486-487
  switches
    /MSG, 487
  write-protect, 250-251
commands, 594
  A (append lines),EDLIN program, 384, 386
  APPEND, 79, 81, 182, 199-208, 229, 327, 344, 347-348, 598-602, 730, 753, 783, 786, 795, 797
  ASSIGN, 79, 83, 133, 144, 159, 206, 221, 333, 344-346, 370, 600-604, 608, 614, 630, 632-633, 659-660, 665, 683, 685, 692, 700, 706, 741, 753, 770-772
  ATTRIB, 79, 81, 217, 231, 242-251, 294, 302, 586, 605-607, 671, 749, 785-787, 792, 794-795
  AUTOEXEC.BAT, 224
  BACKUP, 33, 79, 81, 139, 172, 176, 218, 227, 231, 244-245, 287-296, 299, 302, 344, 347-348, 433, 565, 604, 607-616, 623, 634, 658, 682, 693, 752-753, 755, 770, 772, 784-785, 792, 795, 797
  Batch, 616-618
  BREAK, 78, 84, 625
  BUFFERS, 84, 797
  C (copy lines), EDLIN program, 384, 387-389, 397
  CALL, 78, 83, 415
  CD, 404, 627-629, 826
  CHCP, 78, 82, 85, 303-304, 321-322, 332, 498, 507, 524, 625-627, 715, 718, 722, 727-728, 795
  CHDIR, 59, 78, 81, 181, 191-194, 223, 225, 229, 344, 347-348, 627-629, 659, 825
  CHKDSK, 30, 79-80, 122, 129, 139, 141, 145, 165-172, 176, 179, 189, 213, 293, 299-300, 334, 492, 585-586, 588, 593-594, 597, 604, 629-638, 663, 666, 669, 693-695, 701, 755, 769-771, 777-778, 797, 828
  CLS, 78, 83, 333-334, 370, 411, 638, 639
  COMMAND, 79, 82, 304, 327-329, 332, 415, 639
  command interpreter, 85-87
  command line, 85-87
  COMMAND.COM, 224, 293, 326
  COMP, 79, 82, 157, 169, 231, 263-268, 302, 640-643, 661, 677
  conditional execution, 622-623

CONFIG.SYS, 224
COPY, 30, 57, 78, 82, 139, 146, 149-157, 160-161, 176, 218, 225, 228, 231, 246, 272-286, 302, 338, 379-380, 401, 405-406, 409, 440, 544-545, 557, 617, 620, 631-632, 634, 642, 652-655, 661, 666-667, 669, 784, 787, 825-826
COPY CON, 106, 401, 405, 407, 418, 456, 826-827
COUNTRY, 85, 795, 797
country information management, 85
CTTY, 78, 83, 326, 333, 353-354, 370, 656
D (delete lines), EDLIN program, 380, 384, 389
DATE, 76, 78, 82, 303, 305-308, 331, 446-447, 565, 617, 657-658, 753, 760, 792, 795, 828
DEBUG, 2, 177
DEL (ERASE), 78, 82, 658, 671, 797
DEVICE, 84
DIR, 30, 78, 82, 93, 141, 145-146, 171, 186, 207-208, 228, 234, 254-255, 277, 282-283, 285, 304, 313-318, 331, 365, 401, 585, 588, 591, 595, 604, 658-659, 693-695, 701, 768, 777-778, 788-789, 798
directory management, 81, 181-209, 211-229
disk management, 80-81
disk-level, 121
DISKCOMP, 79-80, 121, 157-160, 179, 308, 345, 347-348, 565, 604, 642, 660-665, 667, 693, 770, 772, 794-795
DISKCOPY, 17, 79-80, 121, 148-157, 177, 179, 228, 287, 345, 347-348, 565, 604, 631-632, 642, 661-669, 682, 693, 761, 770, 772, 794-795
DISPLAY.SYS, 85
DOSSHELL, 79, 82, 304, 318-319, 329, 332, 552
E (end edit), EDLIN program, 379, 382, 384, 387, 397
ECHO ON/OFF, 78, 83, 402-404, 424, 540, 760
ERASE (DEL), 231, 246, 256-260, 300-302, 409, 440, 544-545, 658, 670-671, 825
EXE2BIN, 79, 672-673
EXIT, 78, 82, 304, 328-329, 673
external, 24, 77-85, 122, 205, 208, 225, 234, 263-265, 309, 335
FASTOPEN, 79, 84, 673-675, 795, 798
FC, 79, 82, 157, 169, 231, 268-271, 302, 676-679
FCBS, 84, 765
FDISK, 79-80, 121-126, 138, 178, 344, 347-348, 770, 772, 796, 798
file management, 81-82, 231-302
FILES, 84, 592
FIND, 79, 83, 92, 115, 333, 364, 366-368, 370, 679-681
FOR, 78, 83
FORMAT, 45, 47, 79-80, 93, 121, 125-144, 150-151, 156, 178, 224-225, 308, 344, 347-348, 520, 565, 592, 597, 604, 609, 611-612, 623, 634, 663, 665, 669, 681-688, 693, 701, 758-759, 761, 770, 772, 792, 794, 796, 798, 823-824
GOTO, 78, 84
GRAFTABL, 79, 85, 433, 519, 524, 623, 688-690, 792, 796, 798
GRAPHICS, 79, 83, 333, 335-338, 370, 690-691, 792, 798
I (insert lines), EDLIN program, 375-377, 384, 389, 397
IF, 78, 84
INSTALL, 84
internal, 24, 77-85, 122, 205
JOIN, 79, 83, 159, 221, 333, 344-347, 370, 590, 608, 614, 630, 632, 659-660, 663-664, 667, 683, 691-695, 702, 706, 730, 741, 753, 770-772, 793
KEYB, 79, 85, 333, 350-352, 370, 433, 498, 695-700, 721, 760, 792, 796, 798
KEYN, 623
L (list lines), EDLIN program, 377-378, 382, 385, 388, 390-393, 397
LABEL, 79, 81, 121, 140-144, 178, 344, 347-348, 700-702, 770, 772, 792-793
LASTDRIVE, 84, 604, 770-771
LINK, 2, 585
M (move lines), EDLIN program, 385, 392, 397
MEM, 79, 82, 303, 309-312, 331, 461, 702-705, 797
MKDIR (MD), 59, 78, 81, 181, 186-190, 229, 344, 347-348, 667, 705-706, 770-771, 825
MODE, 79, 82, 303-304, 322-326, 326, 332-333, 354-362, 370, 498, 512-513, 626, 719-726, 796, 798
MODE /STATUS, 517-518
MODE CODEPAGE, 717-718
MODE CODEPAGE /STATUS, 513
MODE CODEPAGE PREPARE, 509-511, 512-517, 523, 626, 698-699, 712-715
MODE CODEPAGE PREPARE SELECT, 512
MODE CODEPAGE REFRESH, 513, 517, 716
MODE CODEPAGE SELECT, 512-513, 516-517, 698, 714-715, 718
MODE COM PORT, 710-712
MODE DISPLAY, 708-709
MODE KEY REPEAT, 718-719
MODE LPTPORT, 706-708, 711-712
MODE REDIRECTION, 711
MORE, 79, 83, 262, 333, 364-367, 370, 726-727
NET PRINT, 733
NLSFUNC, 79, 85, 498, 626-627, 727-729, 795
notation, 580
P (page), EDLIN program, 385, 390-391, 397
parameter, 85-88
PATH, 76, 78-79, 81, 93, 122, 181, 194-201, 225-226, 229, 288, 292, 319, 329, 344, 347-348, 374, 590, 601-602, 612, 614, 729-731, 760, 763, 770, 792
PAUSE, 78, 84, 540
peripheral device management, 333-370
PRINT, 79, 83, 333, 338-342, 380, 731-737
PRINTER.SYS, 85
PROMPT, 76, 78, 82, 93, 187, 288, 300, 303-304, 319-321, 328-329, 332, 420-423, 428-429, 737-740, 763
Q (quit edit), EDLIN program, 379, 381-382, 385, 392-393, 397

R (replace text), EDLIN program, 385, 394-397
RECOVER, 79, 81, 122, 172-179, 189, 586, 671, 740-743
RENAME (REN), 78, 82, 84, 231, 251-256, 302, 583, 743-744
REPLACE, 79, 81, 121, 160-165, 179, 287, 433, 623, 744-751, 794, 798
RESTORE, 33, 79, 82, 139, 172, 218, 231, 244-245, 287, 289-290, 296-302, 344, 347-348, 433, 565, 601, 604, 623, 634, 693, 751-757, 770, 772, 792, 796
RMDIR (RD), 59, 78, 81, 182, 220-223, 229, 344, 347-348, 634, 670, 693, 757-758, 770-771
S (search text), EDLIN program, 385, 393-397
SELECT, 79, 81, 758-763, 792, 794, 798
SET, 78, 83, 93, 122, 187, 196-198, 200-201, 288, 304, 329-332, 446, 593, 730, 737, 763-764
SHARE, 79, 84, 132, 484, 764-766, 792
SHELL, 84, 589, 593, 793-794
SHIFT, 78, 84
SORT, 79, 83, 113-116, 333, 364, 368-370, 766-769
STACKS, 84
stopping, 93
SUBST, 79, 83, 133, 144, 159, 221, 333, 344, 346-350, 370, 590, 604, 608, 614, 630, 632-633, 650, 660, 663-664, 667, 683, 685, 692, 730, 741, 753, 769-772, 793
SUBST, 706
syntax, 185-186
SYS, 79, 81, 121, 132, 145-148, 162, 164, 179, 585, 595, 754, 756, 773-775, 798
system configuration management, 84
system information, 83, 303-332
T (transfer lines), EDLIN program, 385, 396
TIME, 76, 78, 83, 303, 305, 308, 331, 446-447, 565, 591, 617, 657, 753, 760, 775-776, 795, 798, 828
  syntax, 307
TREE, 79, 81, 139-141, 145, 177, 182, 208-209, 211-215, 226, 229, 701, 776-778, 793, 798
TRUENAME, 797
TYPE, 78, 82, 169, 177-178, 231, 240-242, 260-263, 284-286, 302, 379-380, 419, 421, 437-438, 741, 769, 778-779
upper- and lowercase letters in, 581
VER, 78, 83, 303, 308-309, 331, 760, 779
VERIFY, 78, 82, 231, 286-289, 302, 609, 779-780
VOL, 78, 81, 121, 144-145, 178, 781
W (write lines), EDLIN program, 385-387, 397
XCOPY, 33, 79, 81, 139, 149, 157, 182, 215-219, 229, 244-245, 247, 249-250, 287, 294, 308, 400, 433, 520, 540, 544, 667, 759, 761, 781-789, 794
COMn (device name), 103, 235
COMP command, 79, 82, 157, 169, 231, 264-265, 268, 302, 640-643, 661, 677
  compare two text files, 266-267, 271
  syntax, 263-264
COMPAQ, 2, 70, 309
  DOS, 13-15
comparing
  contents of entire disk, 157-160
`Compare error at offset xxxxxxxxx` message, 642
`Compare error on side xx track yy` message, 160
`Compare error(s) on Track tt, side s` message, 662
comparing
  diskettes, 660-664
  files, 676-679
    (COMP), 263-268, 271
    (FC), 268-271
`Compare process ended` message, 662
`Comparing x sectors per track, s side(s)` message, 662
computer
  configuring, 455-495
  layers, 69-72, 74-75
COMSPEC environmental variable, 132, 587, 763
`COMy bbbb,p,d,s,t, initialized` message, 721
CON (device name), 103, 235
concatenation, 272
  COPY command, 273-276
  file and device, COPY command, 283-285
  text files, COPY command, 281
CONFIG.SYS file, 5, 76, 97, 104, 116, 165, 224-226, 260, 262, 284, 306, 325, 327, 331, 348, 373, 383, 404, 418, 450, 456-462, 468, 493-495, 497-498, 507, 509, 514-515, 518-521, 582-586, 588-594, 596, 604, 626-627, 643, 646, 657, 675, 679, 698, 721-722, 758-761, 765, 770-771, 775-776, 789, 796, 826-828
BUFFERS parameter, 107
changes between versions
  2.x and 3.0, 791
  3.1 and 3.2, 793
  3.2 and 3.3, 794
  3.3 and 4.0, 796
DEVICE parameter, 107
directives, 456-494
  international, 500-507, 522-523
  SHELL, 582
  version 4, 489-492
FILES parameter, 107
function, 106-108
  loading utilities into, 490-491
  remarks, 490, 650
  settings, 321
  SHELL parameter, 107
  starting programs from, 649
configuration, 3, 5, 19, 455-495
  checking (MEM), 309-313
  choosing for SELECT utility, 566-567
  commands, 19
  international, 106
  subcommands
    BREAK, 643
    BUFFERS, 644-645
    COUNTRY, 645

DEVICE, 646
DRIVPARM, 646
FCBS, 648-649
FILES, 649
INSTALL, 649
LASTDRIVE, 650
REM, 650
SHELL, 651
STACKS, 652
Configuration too large message, 584
Confirm on Delete (DOS Shell) option, 561
Confirm on replace (DOS Shell) option, 562
console
changing, 656
display mode, 323
Content of destination lost before copy message, 655
CONTROL.nnn file, 612
controller, 99-100
Convert lost chains to file (Y/N) message, 30, 168, 170-172, 334
COPY command, 30, 57, 78, 82, 139, 146, 149-150, 157, 160-161, 176, 218, 225, 228, 231, 272-276, 278, 302, 338, 379-380, 401, 405-406, 409, 440, 544-545, 557, 617, 620, 631-632, 634, 642, 652-655, 661, 666-667, 669, 784, 787, 825-826
read-write attribute, 246
switches, 273-274, 276
/A, 273-277, 653-655
/B, 273, 276-277, 285-286, 653-655
/V, 273, 275, 287, 406, 653, 825-826
syntax, 273
COPY CON command, 401, 405, 407, 418, 456, 826-827
Copy File (DOS Shell) panel, 558
copy-protected files, 299
Copy... (Dos Shell) option, 538, 546, 550, 557-558
copying
archived files switch, XCOPY, 218, 245, 247, 249-250
directory management (XCOPY), 215-219
concatenating
file and device, 283-285
text files, 281
contents of disk, 148-157
data from floppy, 408-410
device to file, 283
directory
date switch, XCOPY, 218
to directory, 277-279
disk to disk, 280-281
diskettes, 663-669
join files, 285-286
files, 413, 652-654
(COPY), 272-286
then erasing, 441
from subdirectories, 781-789
lines with EDLIN, 384, 387-389
modified files switch
BACKUP, 245
XCOPY, 217-218, 245, 247, 249-250
system files, 145-148
to and from logical devices, 272-286
country code, 501, 657-658
COUNTRY configuration subcommand, 85, 456, 498, 500-504, 509, 521, 585, 589, 645, 696, 728, 782, 791, 795, 797
country information management, 80
commands, 85
COUNTRY.SYS device driver, 484, 502-504, 507, 626, 727-728, 759, 761-762, 795
CP/M operating system, 233
CPI file extension, 236, 515
CPU (central processing unit), 21-22, 71-72, 74-75, 94
expanded memory, 111-113
extended memory, 111
function, 73
Create directory... (DOS Shell) option, 561
CRT (cathode ray tube), 22
Ctrl key, 90-91
Ctrl-@ (null) keys, 384
Ctrl-Alt-Del keys, 91, 107, 813, 818, 824
Ctrl-Alt-F1 keys, 512
Ctrl-Alt-F2 keys, 512
Ctrl-Break keys, 91, 93, 101, 156-157, 261-262, 301, 358, 365, 376-378, 403, 456-457, 616, 623, 625, 643, 813
Ctrl-C (^C) keys, 91, 93, 134, 261-262, 301, 376-378, 389, 403, 441, 456-457, 623, 643, 813
Ctrl-G (^G) keys, 234
Ctrl-I keys, 768
Ctrl-Num Lock keys, 91, 262, 813
Ctrl-P keys, 813
Ctrl-PrtSc keys, 91, 444, 778, 813
Ctrl-S (^S) keys, 234, 240-242, 262, 391, 444, 813
Ctrl-V key, 383
Ctrl-Z (^Z) keys, 241-242, 261, 264, 267, 272, 275-277, 285-286, 302, 381, 383-384, 396
CTTY command, 78, 83, 326, 333, 354, 370, 656
syntax, 353
current
directory, 62, 186-199, 205, 215, 220-226, 317, 346-347
disk, 88
drive, 62, 192-193, 195, 209, 215
full path, 187
path, 321, 332
Current CON code page: codepage message, 699
Current drive is no longer valid message, 584
Current keyboard code: keycode code page: codepage message, 699
Current keyboard does not support this Codepage message, 721
cursor
keys, 89
ANSI.SYS keyboard scan codes, 835

menu selection in DOS Shell, 534
movement in ANSI.SYS file, 421-422
position, 829-830
customizing
DOS, 824-828
DOS Shell, 569-574
prompt, 737-740
cyclic-redundancy check (CRC), 131
cylinder, 36-40, 46-47, 52-53, 62-65, 123, 128, 149

D

/D switch
BACKUP command, 291
JOIN command, 345-346, 692
REPLACE command, 161-163
SUBST command, 770, 772
/D:date switch
BACKUP command, 607, 614
XCOPY command, 215, 217-219, 782, 785
/D:ddd switch
DRIVER.SYS device driver, 474-475
/D:device switch
PRINT command, 339-342, 731
/D:num switch
DRIVPARM configuration subcommand, 647
/DATE (DOS Shell) option, 573
/DEBUG switch
MEM command, 310, 313, 703-705
/DOS (DOS Shell) option, 570
D (delete lines) command, EDLIN program, 380, 384, 389
d:path\filename.ext - File not found message, 642
d:path\filename.ext - File sharing conflict message, 642
d:path\filename.ext - Invalid path message, 643
DAT file extension, 236
data
bus, 74
checking accuracy of, 779-780
file searching (APPEND), 200-208
format, 39-40, 67
organization, 48-67
storage, 48-67
Data message, 596
DATE command, 76, 78, 82, 303, 305, 307-308, 331, 446-447, 565, 617, 657-658, 753, 760, 792, 795, 828
syntax, 306
dates
displaying from DOS Shell, 573
Personal Computer AT, 828
system, 657-658
dBASE III Plus, 227
dBASE IV, 6, 169, 227, 246
DBF file extension, 236
DCT file extension, 236
DEBUG program, 2, 177, 225, 635
defaults, 88, 215
changing command processor, 651
directory, 61-62, 195, 199
disk, 88
names, 227
number of lines, FC command, 270
value, read-write attribute, 243-244, 246
VERIFY setting, 287
DEL (ERASE) command, 78, 82, 658, 671, 797
/P switch, 797
Del key, 89, 92, 814
EDLIN, 384
Delete current volume label (Y/N)? message, 702
Delete Program (DOS Shell) panel, 546
Delete... (DOS Shell) option, 538, 546, 549, 559
DELETE.BAT file, 406-407
delimiter, 86
Deselect All (DOS Shell) option, 561
destination
device, COPY command, 274-275
disk, 664
file, COPY command, 215, 274-275, 653-654
specifier, 86
name, 216
parameter, 86
DEV file extension, 236
DEVICE configuration subcommand, 84, 456, 461-462, 468, 586, 594, 646, 794
device drivers, 76, 98, 100-101, 106-108, 226, 582, 596, 646
ANSI.SYS, 319-321, 325-326, 482, 646, 797
BIOS (basic input/output system), 286
block, 101-102, 116
character, 100-104, 116
COUNTRY.SYS, 484, 759, 761-62
disk drive, 100-101
display adapter, 99
DISPLAY.SYS, 480-82, 498, 500, 502-504, 509, 514-515, 518, 626, 713, 718, 721-722, 794, 798
DRIVER.SYS, 474-479, 793
error messages, 581, 595-598
expanded memory, 109, 111-113
IBMCACHE.SYS, 462
inconsistent, 594
incorrect header length, 596
installing, 102, 104-106, 417-418, 461-463, 468
keyboard, 351-352
KEYBOARD.SYS, 484, 511-512, 759, 761-762
logical device, 102, 105-106
parameters incorrect, 586
PCIBMDRV.MOS, 570
PCMSDRV.MOS, 570
PCMSPDRV.MOS, 570
PRINTER.SYS, 480, 482-484, 498, 500, 502, 504-507, 509, 514-515, 518, 626, 713, 718, 721-722, 794, 798

RAMDRIVE.SYS, 469-471, 478-479, 646
resident, 102-104, 116
SMARTDRV.SYS, 462, 468
software, 70
Basic Input/Output System (BIOS), 70
unable to use device, 596
VDISK.SYS, 101-102, 469, 791, 799
XMA2EMS.SYS, 479, 796
XMAEM.SYS, 479-480, 796
`Device error during Prepare` message, 722
`Device error during Refresh` message, 722
`Device error during Status` message, 722
`Device error during write of font file to device` message, 722
device name
AUX, 235, 340, 353
COM1, 340, 353-354, 358
COM2, 340, 353, 358
COM3, 340, 353, 358
COM4, 340, 353, 358
COMn, 235
CON, 235
LPT1, 340, 342
LPT2, 340
LPT3, 340
LPTn, 235
NUL, 235
PRN, 235, 340, 342
`Device or codepage missing from font file` message, 722
devices, 97-99
code pages, 714-715, 717-718
reloading, 716
control codes, 240-242
could not write data to, 597
destination, COPY command, 274-275
display information, 719, 726-727
input/output, 114
keyboard input, 89-92
level, 98
logical, 272
managing, 24-26
mode, 322-323
names, 235
not ready, 597
output, 89
parameter, 235
PRN, 380
redirecting, 114
refreshing memory, 517
set mode of operation, 719-725
source, COPY command, 274-275
status report, 830
substitution
piping, 115-117
redirection, 114-115
DIF file extension, 236
DIR command, 30, 78, 82, 93, 141, 145-146, 171, 186, 207-208, 228, 234, 254-255, 277, 282-283, 285, 304, 314-318, 331, 365, 401, 585, 588, 591, 595, 604, 658-659, 693, 694-695, 701, 768, 777-778, 788-789, 798
piped command, 116
switches
/P, 313-316, 658-659
/W, 313, 316, 658-659
syntax, 313
directives
adding international
AUTOEXEC.BAT file, 523-524
CONFIG.SYS file, 522-523
BREAK, 456-457
BUFFERS, 456, 458, 491, 795
CHCP, 507-509, 512, 516-517
CONFIG.SYS file, 456-494
version 4, 489-492
COUNTRY, 456, 498, 500-504, 509, 521, 585, 589, 696, 728, 782, 791
DEVICE, 456, 461-462, 468, 586, 594, 794
FASTOPEN, 491-493
FCBS, 456, 484, 596, 791
FILES, 456, 485, 585, 789
improperly phrased, 595
INSTALL, 456, 490-491, 796
KEYB, 507
LASTDRIVE, 456, 485-486, 791
MODE, 507
NLSFUNC, 507-509
REM, 457, 490, 796
SHELL, 457, 486-488, 590-591, 640
STACKS, 457, 488-489, 588, 592, 793
SWITCHAR, 791
SWITCHES, 457, 490, 796
directories, 53, 166
accessing from DOS Shell, 552-554
alias files, 189, 191, 194, 198-199, 223
allocating memory to store information, 491-493
changing, 627-629
current (CHDIR), 191-194
checking, 629-638
status with CHKDSK, 165-168
copying, COPY command, 277-280
creating from DOS Shell, 561
current, 62, 186-199, 205, 215, 220-226, 317, 346-347
date, 54
field, 183
default, 61-62, 132, 199
deleting (RMDIR), 221-223
displaying files, 776-778
erasing files, 257-259
file
attribute, 54, 183
name, 54, 183, 252-256
extension, 54, 183
path name, 61, 187-189

searching for, 598-602, 729-731
size, 183
hierarchy, 58-62, 67, 181-186, 209, 211-213, 224-225, 229, 313, 346-350, 730
keeping files, 228-229
listing, 658-659
making (MKDIR), 187-190
management, 80-81, 183
naming, 189-190
organization, 58-62, 66-67
parent, 317
path, 61-62, 593
rebuilding with RECOVER, 176-178
removing (RMDIR), 220-223
root, 128, 149, 182-190, 209, 220-222, 224-226, 294, 329, 346-347
search path, 198-199
subdirectory, 183, 228-229, 346-350
deleting, 220-223
displaying directory, 318
DOS files, 226
sorting, 369-370
temporary, 227-228
time, 54, 183
Directory Count Overflow message, 777
Directory not empty message, 695
directoryname Cannot recover direntry entry, processing continued message, 634
directoryname Convert directory to file (Y/N)? message, 635
directoryname Invalid current directory message, 637
directoryname is joined, tree past this point not processed message, 635
Disk boot failure message, 584
disk caches, 462-468, 644
Disk Compare (DOS Shell) option, 565
Disk Copy (DOS Shell) option, 565
disk drive, 22, 30, 32-34, 37, 67, 125, 128, 602-604
alignment, 38
batch file to prevent formatting, 438-439
capacity, 43
controlling additional, 474-478
current, 192-193, 195, 209, 215
cylinder, 36-38
driver program, 32
floppy, 22, 33-35, 42-46, 58, 132-133
hard disk, 22, 32-33, 35, 42-47, 58, 132-133
joining with subdirectory, 691-695
keeping track of, 571
logical, 64-67, 101, 123, 125, 132, 474, 486
maximum number attached, 485-486
name, 184-185, 478-479, 590
physical drive, 123, 125, 474
SELECT utility, 567-568
setting last valid, 650
track, 35-38, 597
unable to find, 584, 587
Disk error reading FAT x message, 635
Disk error writing FAT x message, 635
Disk full error writing to BACKUP Log File Strike any key to continue message, 613
disk operating system *see* DOS
Disk unsuitable for system disk message, 685
Disk-error-type error on file filename message, 737
DISKCOMP command, 79-80, 121, 157, 159, 179, 308, 345, 347-348, 565, 604, 642, 660-665, 667, 693, 770, 772, 794-795
error messages, 160
switches
/1, 158, 660-661
/8, 158, 660-661
syntax, 158
DISKCOMP.COM file, 413
DISKCOPY command, 17, 79-80, 121, 148, 150-152, 177, 179, 228, 287, 345, 347-348, 565, 604, 631-632, 642, 661-669, 682, 693, 761, 770, 772, 794-795
dual disk drives, 153-154
error messages, 156-157
high-capacity (HC) disk/drive, 156
one disk drive, 154-156
switches
/1, 149, 664-665
syntax, 149
DISKCOPY.COM file, 413
diskettes, 34, 136
see also disks and floppy disks
3 1/2-inch, 34, 43-45, 49
device driver, 103
DISKCOPY command, 149-152
FORMAT command switch, 127
root directory capacity, 130
5 1/4-inch, 34, 43-45, 49, 136
DISKCOPY command, 149-152
FORMAT command switch, 127
8-inch, 33-34
checking status, 165-172
comparing, 157-160, 436-437, 660-664
copying, 148-157, 663-669
DISKCOPY
source disk, 150-151
target disk, 150-151
dismantling and rebuilding disk, 172-173, 178
external DOS commands, 80
formatting, 133-136
with DOS Shell, 137-138
labeling, 144-145
transferring system files, 148
updating selected files, 160-165
write protect, 156, 597-598
Diskettes compare OK message, 662
disknumber diskette bad or incompatible message, 662
disks
address, 123
boot, 16-17, 107-108, 225

cold, 107
record, 128, 131
warm, 107
buffers, 458-461
capacity, 39-40, 43-49
changing, 590
checking
directory and file allocation table (FAT), 629-638
status, 165-172
volume label, 145
cluster size, 49
control area, 128, 130-131
copying files from another disk, 744-751
controller, 100
cluster size, 49
default, 88, 132
defective, 596-597
density, 45-47
destination, 216, 664
directories, 53-67, 209
rebuild with RECOVER, 176-178
root, 58-62, 67, 130-131, 168, 184-186
subdirectory, 58-62, 130, 168, 184-186
displaying
subdirectories, 776-778
volume label, 781
error detected, 595
files, 31-34, 128, 131
fixed, 30
floppy, 124, 136
see also floppy disks
compare contents, 157-160
copy, 148-157
DISKCOPY, source and target disk, 150-151
external DOS commands, 80
formatting with DOS Shell, 137-138
labeling, 144-145
transferring system files, 148
write protect, 156
formatting, 41-47, 52-53, 130
fragmentation, 129, 139, 165-166
full, 588
hard, 62, 313
changing command prompt, 319
disk formats, 47
external DOS commands, 79
formatting, 138-141
increasing performance, 458-461
initialize, 128, 130-141, 681-688
Install, 565-566
logical structure, 149
magnetic storage, 31
maintaining, 121-179
management, 80-81, 128-179
naming, 141-144
nonbootable, 593
optimizer, 129
partitioning, 62, 122-126, 130, 138, 764-766
boot record, 128
deleting, 125-126
extended, 64-65, 125
logical, 132
primary, 64-65, 125
table, 62-65, 67, 124
RAM, 463-464, 826-827
random access, 460
removable, 30
sectors, 41-42
sequential reads and writes, 460
source, 664
subdirectory, 172
damaging to entries, 166
repairing with RECOVER, 176-178
system files, 16-17
tracks, 35-42, 62
transferring system files, 145-148
unformatted, 586
updating selected files, 160-165
virtual, 469
volume label, 700-702
display, 5, 99-100
column control, 323
erasing, 830
international characters, 480-482
messages, 410
modes of DOS Shell, 529-531, 561
number of lines and columns on screen, 325
range of line numbers with EDLIN, 377
text with EDLIN, 377-378
text file (TYPE), 260-263
display adapters, 22, 99-100
Color Graphics Adapter (CGA), 99, 519, 529, 708-709
Enhanced Graphics Adapter (EGA), 99, 105, 529, 709
Liquid Crystal Display (LCD), 105
Monochrome Display Adapter (MDA), 99, 529, 709
switching between monochrome and graphics, 708-709
Video Graphics Array (VGA), 99, 529-530, 708-709
DISPLAY.SYS device driver, 85, 105-106, 480-482, 498, 500, 502-504, 509, 514-515, 518, 626, 713, 718, 721-722, 794, 798
parameters, 481
DisplayWrite 4, 556
`Divide overflow` message, 584-585
`Do you see the leftmost □? (Y/N)` message, 324, 723
`Do you see the rightmost ٩? (Y/N)` message, 723
DOC file extension, 237, 556, 573
`Does %s specify a file name or directory name on the target (F=file, D=directory)?` message, 786
`Does WORDS specify a filename of directory name on the target (F=file, D=directory)?` message, 216
DOS (disk operating system), 12, 14-19, 233

> prompt, 17, 24, 69, 76, 93
booting, 75-76
configuration, 19
customizing, 824-828
disk buffers, 459-461
display version, 331, 779
environment, 93-94
error while loading, 585
files, 16-17, 27-67
installation utility, 758-763
interface, 303-306, 332
international
directives, 500-507
features, 497-525
kernel, 23, 25, 72, 98-102, 104-108, 204
master disks, 151-152
parameters, 19
placing copy on disk, 773-775
switching to DOS Shell prompt, 572
unable to load, 584
version 1.0, 127
version 1.1, 127
version 2.0, 229
version 2.x and 3.0
changes between, 791-792
version 3.0, 229, 374
version 3.0 and 3.1
changes between, 793
version 3.1 and 3.2
changes between, 793-794
version 3.2 and 3.3
changes between, 794-796
version 3.3
attribute settings, 244
BACKUP command, 291-293
restore with earlier version, 299
data file search (APPEND), 200-202, 204-206, 208
DATE command, 307-308
GRAPHICS command, 335
KEYB, 350-352
listing directories (TREE), 209, 211
MODE command, 359-360
RESTORE command, 297-298
upgrade with REPLACE command, 163-165
version 3.3 and 4.0, 2
CALL subcommand, 442
changes between, 796-799
FDISK program, 820-823
KEYB command, 510-512
version 4.0, 5, 127
/V switch, 133
adjust the display mode (MODE), 326
ASSIGN command, 344
BACKUP command, 291-293, 295-296
BUFFERS directive, 461
data file search (APPEND), 200-201, 203-206, 208
DATE command, 307-308
device driver, 103, 105-106
DISKCOPY command, 152-156
DOS Shell, 26, 303, 318-319, 332
ERASE command pause switch, 258
expanded memory, 479-480
FORMAT, 134-135
hard disk, 139
with DOS Shell, 137-138
GRAPHICS command, 335, 337
installing with SELECT utility, 565-569
KEYB, 350-352
listing directories (TREE), 209, 211
memory use, 108-109
MEM command, 309
MODE command, 325, 354-360
repair allocation size error, 170-172
REPLACE command, 160
SELECT utility, 519
SORT filter, 369-370
SYS command source drive spec, 146-147
TIME command, 307-308
upgrade with REPLACE command, 165
volume label, 131
DOS Shell, 5, 24, 26, 90, 224-225, 527-575
accessing directories, 552-554
adding programs, 538-548, 550
associating files, 556-557, 573
changing, 572
file attributes, 559-560
programs, 545-546, 548-549
color display, 564
COM2 port for mouse, 574
copying
files, 557-559
programs, 546-547, 550-551
customizing, 569-574
deleting
files, 559, 561
programs, 546, 549
directories, 561
display, 529-531
date and time, 573
exiting, 529, 572
formatting diskettes, 137-138
function keys, 536
Help feature keys, 815
help system, 532-533
installing with SELECT utility, 565-569
limiting memory for buffers, 573
menus
Arrange, 563-564
DOS Utilities, 565
Exit, 536
File System, 561-563, 570-571
Group, 535
Index, 533
Main Group, 529, 552-564
customizing, 549-551
Program, 535

pull-down, 534-535
Start Programs, 529, 534-536
Start Programs/Main Group, 527
tree, 527-529
moving files, 557
multiple windows, 563-564
opening files, 554-555
options
/ASC, 573
/B, 573
/CLR, 571-572
/CO1, 573-574
/CO2, 573-574
/CO3, 573-574
/COLOR, 570
/COM2, 574
/DATE, 573
/DOS, 570
/EXIT, 572
/LF, 574
/MAINT, 572
/MENU, 571
/MEU, 571
/MOS:, 570
/MUL, 571
/PROMPT, 572
/SND, 571
/SWAP, 572
/TEXT, 574
/TRAN, 570
Add..., 535, 537-538, 550-551
Associate..., 556-557
Backup Fixed Disk, 565
Change..., 537, 546, 548
Command Prompt, 529, 552
Confirm on Delete, 561
Confirm on replace, 562
Copy..., 538, 546, 550, 557-558
Create directory..., 561
Delete..., 538, 546, 549, 559
Deselect All, 561
Disk Compare, 565
Disk Copy, 565
Display options, 561
Exit, 534, 564
File options, 561
Format, 565
Group, 538
Multiple File List, 563
Open..., 555
Print..., 555
Rename..., 559
Reorder..., 549
Restore Fixed Disk, 565
Select across directories, 562
Select All, 558, 561
Set Date/Time, 565
Shell Change attribute..., 559-560
Start, 537-538
System file list, 564
View, 560
printing files, 555-556
renaming files, 559
scrolling text, 531-532
selecting all files, 561
special keys, 814-815
starting programs, 537-538, 552
submenus
Exit, 552
Group, 547-551
Program, 537-547, 550
switching to DOS prompt, 572-
viewing files, 560
DOS Utilities (DOS Shell) menu, 565
DOSSHELL command, 24, 79, 82, 304, 318, 329, 332, 552
syntax, 319
DOSSHELL.BAT file, 551, 568-571, 573
dot (.) current directory name, 188-189, 220, 222-223
dot dot (..) parent directory name, 189, 191, 194, 198-199, 220, 222-223, 279-280, 318
`Drive d: not ready Make sure diskette is inserted into the drive and the door is closed` message, 663, 667
`Drive letter must be specified` message, 685
`Drive not ready` message, 596
`Drive or diskette types not compatible` message, 662-663
`Drive types or diskette types not compatible` message, 667
DRIVER.SYS device driver, 105, 474-479, 793
/C switch, 476
/D:ddd switch, 474-475
/F:f switch, 475
/H:hh switch, 475
/N switch, 476
/S:ss switch, 475
/T:ttt switch, 475
DRIVPARM configuration subcommand, 647
/D:num switch, 647
/F:type switch, 647
/H:hds switch, 647
/I switch, 647
/N switch, 647
/S:sec switch, 647
/T:trk switch, 648
DRV file extension, 237
DTA file extension, 237
DUMP.EXE program, 177
`Duplicate filename or File not found` message, 256, 744

E

/E switch
APPEND command, 200-203, 206, 599-601

IBMCACHE.SYS device driver, 468
RAMDISK.SYS file, 592
RAMDRIVE.SYS device driver, 470
RESTORE command, 298, 796
SHELL directive, 487
XCOPY command, 215-217, 782, 784
/E:eeeee switch
COMMAND command, 327-328
/E:max switch
RAMDRIVE.SYS device driver, 469-471
/E:size switch
COMMAND command, 639
SHELL, 487, 590-591, 793-794
/E:time switch
RESTORE command, 752-753
/EXIT (DOS Shell) option, 572
E (end edit) command, EDLIN program, 379, 382, 384, 387, 397
`ECHO is status ON/OFF` message, 620
ECHO ON/OFF subcommand, 78, 83, 402-404, 410-411, 424, 431-432, 540, 618-620, 623-624, 760
editing keys, 813-814
EDLIN, 382-384
EDLIN (text editor) program, 225, 373-374, 376-380, 401, 418, 421, 428-429, 456, 619
* prompt, 375-378, 380, 382, 387
? prompt, 393
adding end-of-file character, 383
appending lines, 384, 386
canceling changes, 382
creating
CONFIG.SYS, 106
memo, 374-375
commands
A (append lines), 384, 386
C (copy lines), 384, 387-389, 397
D (delete lines), 380, 384, 389
E (end edit), 379, 382, 384, 387, 397
I (insert lines), 375-377, 384, 389, 397
L (list lines), 377-378, 382, 385, 388, 390-393, 397
M (move lines), 385, 392, 397
P (page), 385, 390-391, 397
Q (quit edit), 379, 381-382, 385, 392-393, 397
R (replace text), 385, 394-397
S (search text), 385, 393-397
T (transfer lines), 385, 396
W (write lines), 385-387, 397
copying lines, 384, 387-389, 397
current line, 379
deleting
lines, 380, 384, 389
text, 380
editing, 379, 381-385, 387, 392-393, 397
keys, 383-384
file extensions, 381-382
graphics characters, 383
inserting, 375-378, 384, 389, 397
line numbers, 376-380
list
lines, 377-378, 382, 385, 388, 390-393, 397
text, 377
moving lines, 385, 392, 397
page lines, 385, 390-391, 397
printing text, 379
prompt, 379, 386, 389, 394-396
replacing text, 385, 394-397
saving text, 379
searching text, 385, 393-397
special key combinations, 383-384
switches
/B switch, 381
syntax, 385, 387-388, 392, 394-396
EGA (Enhanced Graphics Adapter), 22, 99, 105, 324-325, 499, 529
ellipsis (...) character, 195
`End of input file` message, 382, 386-387
end-of-file (^Z) character, 240-242, 261, 264, 267, 272, 302, 381, 642, 653
enhanced keyboard, 91, 488-489
`Enter current Volume Label for drive d:` message, 686
Enter key, 89, 101, 134, 376-379, 382, 386-387, 389-392, 394-395, 535, 813-814
environmental variables, 329-331
`EOF mark not found` message, 642
EPROM (programmable ROM), 73
ERASE command, 231, 256, 300-302, 409, 440, 544-545, 658, 670-671, 825
read-write attribute, 246
switches
/P switch, 257, 670, 797
syntax, 257-260
erasing files, 658, 670-671
`Error during read of font file` message, 723
`Error in COUNTRY command` message, 585
`Error in EXE file` message, 585
`Error loading operating system` message, 585
error
memory parity, 175
messages, *see* messages
`Error opening logfile` message, 613
`Error reading partition table` message, 686
`Error writing partition table` message, 686
`Errors found, F parameter not specified Corrections will not be written to the disk` message, 635-636
`Errors in list device indicate that it may be off-line. Please check it.` message, 735
Esc key, 90-92, 382-383, 535, 814-815
EDLIN, 384
escape characters, 419, 421
EXE file extension, 77-78, 194, 200-201, 206, 237, 269, 401, 414, 554, 556, 585, 617, 677, 729, 734
change to BIN or COM files, 672-673
EXE2BIN command, 79, 672-673
EXEC DOS function call, 600
`EXEC failure` message, 585-586

Exit (DOS Shell)
menu, 536, 552
option, 534, 564
exit codes, 433
EXIT command, 78, 82, 304, 329, 673
syntax, 328
Exiting after a fatal error to DOS message, 17
expanded memory (EMS), 105-106, 109, 111-113, 117
page, 111-113
versions
DOS version 4, 479-480
LIM, 311
LIM 3.2, 113, 117
LIM 4.0, 113, 117
extended
addressing, 109, 111
ASCII codes, 806-811
keyboard, 490
memory, 105-106, 109, 111, 117
protected mode, 109, 111, 117
scan codes, 425, 427
search switch
APPEND command, 200-208
external commands, 24, 77-81, 122, 205, 208, 225, 263-265, 309, 335
notation, 580-581

F

/F switch
BACKUP command, 291-293, 295, 608-609, 612-613, 682, 795
CHKDSK command, 166, 586, 629-631, 634-638
TREE command, 139, 209, 212-213, 659, 777, 793
/F:f switch
DRIVER.SYS device driver, 475
/F:name_space switch
SHARE command, 764-765
/F:size switch
BACKUP command, 291, 608-609
FORMAT command, 682, 798
/F:type switch
DRIVPARM configuration subcommand, 647
F1 function key, 90, 92, 533-534, 814-815
EDLIN, 383
Help DOS Shell function key, 536
F2 function key, 92, 546-550, 814
EDLIN, 383
F3 key, 90, 92, 378, 814-815
EDLIN, 383
F4 function key, 92, 540, 814-815
EDLIN, 384
F5 function key, 92, 466, 814
EDLIN, 384
F6 function key, 92, 242, 466, 814
EDLIN, 384
F7 function key
EDLIN, 384
F9 key, 815
F10 function key, 90, 529, 534, 815
Start Programs/Main Group menu DOS Shell function key, 536
F11 function key, 533, 815
Failure to access Codepage Font File message, 723
Failure to access COUNTRY.SYS message, 761
Failure to access device: devicename message, 723
Failure to access KEYBOARD.SYS message, 761
FASTBACK utility, 129
FASTOPEN already installed message, 675
FASTOPEN command, 79, 84, 491-493, 673-675, 795, 798
switches
/X, 492, 674, 798
parameters, 491-492
FASTOPEN installed message, 675
FASTOPEN.EXE file, 312, 490, 649, 796
FC command, 79, 82, 157, 169, 231, 269-271, 302, 676-679
switches
/A, 269, 676
/B, 268, 270, 676-677
/C, 269, 676-677
/L, 269-270, 676
/LBn, 269-270
/LB x, 676
/N, 269-271, 676
/nnnn, 269-270
/T, 270, 676
/W, 270-271, 676-677
/x switch, 676, 678
syntax, 268-269
fc: cannot open filename - No such file or directory message, 678
fc: filename longer than filename message, 678
fc: incompatible switches message, 678
fc: no differences encountered message, 678
fc: out of memory message, 678
FCB unavailable message, 596
FCBS configuration subcommand, 84, 456, 484, 596, 648-649, 765, 791
FDISK command, 79-80, 121, 124-126, 138, 178, 344, 347-348, 591, 770, 772, 796, 798, 817-823
DOS version 3.2 and earlier, 818-819
DOS version 3.3 and 4, 820-823
hard disk partitions, 123
syntax, 122-123
File (DOS Shell) submenu, 554-561
file allocation table (FAT), 48-57, 63, 66-67, 80, 128-131, 141, 149, 151, 158, 161, 165-171, 176, 290, 299, 586, 597, 629-638, 662, 669, 684
directories, 181, 183, 188-189
ERASE command, 257
File allocation table bad drive d: message, 586, 636
File cannot be copied onto itself message, 655, 749, 786-787

file control block (FCB), 484, 648-649
File creation error message, 586, 761, 787
file extensions, 29
.SSS, 375, 381-382
.. (temporary or incorrectly stored file), 237
.ARC, 236
.ASC, 236
.ASM, 236
.BAK, 236, 382, 641
.BAS, 236, 556
.BAT, 77-78, 194, 200-201, 206, 236, 401, 414, 554, 556, 616- 617, 729
.BIN, 236, 269
.C, 236
.CBL, 236
.CFG, 236
.CHP, 236
.CNF, 236
.COM, 77-78, 194, 200-201, 206, 236, 269, 401, 414, 554, 556, 617, 677, 729, 734, 825
.CPI, 236, 515
.DAT, 236
.DBF, 236
.DCT, 236
.DEV, 236
.DIF, 236
.DOC, 237, 556, 573
.DRV, 237
.DTA, 237
.EXE, 77-78, 194, 200-201, 206, 237, 269, 401, 414, 554, 556, 585, 617, 677, 729, 734
.FNT, 237
.HLP, 237
.IDX, 237
.IMG, 237
.KEY, 237
.LET, 237
.LIB, 237, 269
.LST, 237
.MAC, 237
.MAP, 237
.MEU, 237, 548, 550
.MOS, 237
.MSG, 237
.NDX, 237
.OBJ, 237, 269
.OLD, 237
.OVL, 237
.OVR, 237
.PAK, 237
.PAS, 237
.PCX, 237, 556
.PIF, 237
.PRN, 237
.PRO, 237
.PS, 237
.RFT, 237
.SAV, 237
.STY, 237
.SYS, 237, 269, 825
.TIF, 237
.TMP, 237
.TST, 237
.TXT, 237, 641, 744
.WK1, 237
.WK3, 237
.WKQ, 237, 573
.WKS, 237
Basic Input/Output System (BIOS), 76, 107
file names, 29, 53-54, 61, 216, 232-239
changing, 252-256
directory entry, 183
extensions, 29, 54, 61, 183, 232, 236-237
invalid characters, 233-235, 238-239
prefix, 29, 232
reserved names, 235
root, 232, 234, 317
suffix, 29, 232
truncating, 238-239
valid characters, 233
File not found message, 178, 439, 586, 626, 659, 671, 728, 742
File options (DOS Shell) option, 561
File System (DOS Shell) menu, 552-564
File View (DOS Shell) panel, 560
filename Allocation error for file, size adjusted message, 633
filename Contains invalid cluster, file truncated message, 634
filename contains n non-contiguous blocks message, 633
filename Contains xxx noncontiguous blocks message, 634
filename device driver cannot be initialized message, 586
filename File creation error message, 754
filename File not found message, 736, 787
filename File not in PRINT queue message, 736
filename First cluster number in invalid, entry truncated message, 636
filename has invalid cluster, file truncated message, 636
filename is cross-linked on cluster X message, 168
filename is currently being printed filename is in queue message, 736
filename Pathname too long message, 736-737
filename1 Is cross linked on cluster x filename 2 Is crosslinked on cluster x message, 636-637
files, 4
720.BAT, 402-404
ANSI.SYS, 105-106, 383, 417-430, 450-451
archived, 245, 247, 249-250
ASCII, 174, 177, 337, 364, 366-368, 373-374, 381, 400, 418, 456, 617, 653
associating, DOS Shell, 556-557, 573
attributes, 54, 166

changing from DOS Shell, 559-560
control (ATTRIB), 242-251
directory entry, 183
AUTOEXEC.BAT, 24, 76, 108, 122, 156, 165, 194, 196, 225-226, 260, 284-285, 305, 318, 327, 330, 334, 338, 446-449, 490, 493, 497-498, 507-509, 519-521, 523-524, 617, 627, 657, 740, 758-761, 796, 827
backing up, 607-616
BACKUP.nnn, 612
BACKUPID.@@@, 612, 755
BAT file extension, 77-78, 194, 200-201, 206
batch, 77-78, 399-415, 616-618, 795
advanced techniques, 431-451
binary, 169, 174, 381
BRIDGE.DRV, 106
CD.ASC, 404-405
CHKDSK.COM, 413
COM file extension, 77-78, 194, 200-201, 206
COMMAND.COM, 76, 328, 330, 445, 486-487, 582-585, 587, 589, 592, 595, 601, 639, 651, 688, 754, 756, 763, 773-775
comparing, 640-643, 676-679
diskettes or subdirectories, 436-437
CONFIG.SYS, 5, 76, 97, 104, 106-108, 116, 165, 225-226, 260, 262, 284, 306, 321, 325, 327, 331, 348, 373, 383, 404, 418, 450, 456-462, 468, 493-498, 500-507, 509, 514-515, 518-523, 582-586, 588-594, 596, 604, 626-627, 643, 646, 657, 675, 679, 698, 721-722, 758-761, 765, 770-771, 775-776, 789, 791, 793, 796, 826-828
contiguous, 50
CONTROL.nnn, 612
copying, 413, 441, 652-655
from DOS Shell, 557-559
from subdirectories, 781-789
to hard disk, 824-828
COUNTRY.SYS, 502-504, 507, 626, 727-728, 795
cross-linked, 166
DELETE.BAT, 406-407
deleting from DOS Shell, 559
deselecting all from DOS Shell, 561
destination, 215, 653-654
COPY command, 274-275
directory, 53-54, 58-62, 66-67
disk, 30-67
disk drive, 30-40, 67
DISKCOMP.COM, 413
DISKCOPY.COM, 413
dismantling and rebuilding, 172-175, 178
DISPLAY.SYS, 105-106
displaying
attributes, 605-607
contents on-screen, 778-779
from directory, 776-778
strings from, 679-681
DOSSHELL.BAT, 551, 568-571, 573
DRIVER.SYS, 105
EDLIN program, 381-382
embedding escape characters, 419
erasing, 658, 670-671
EXE file extension, 77-78, 194, 200-201, 206
FASTOPEN.EXE, 490, 796
FILExxxx.CHK, 638
FLOPPY.BAT, 413
formatting, 289
FORMAT.BAT, 438
FORMAT.COM, 401, 413, 438
FORMAT.EXE, 401
GRAPHICS.PRO, 337
handle, 18, 485
HOME.ASC, 404-405
HOME.BAT, 404-405
IBMBIO.COM, 102, 107, 116
IBMCACHE.SYS, 106
IBMDOS.COM, 23, 107
intermediate, 375
IO.SYS, 102, 104, 107, 116, 312, 584, 688, 754, 756, 773-775
ISIBM.BAT, 434-435
KEYB.COM, 490, 796
KEYBOARD.SYS, 696, 698
KOALA.SYS, 106
list switch, TREE command, 209, 212-213
locking, 596, 764-766
management, 4, 24-26, 47, 80-81, 231-302
commands, 81-82, 231-302, 319
MARK.BAT, 404-405
MENU.BAT, 449-450
modified
copy, 217-218, 245, 247, 249-250
restore, 245
MORER.BAT, 439
MOUSE.SYS, 106
MOVE.BAT, 440-441
moving from DOS Shell, 557
MSDOS.SYS, 23, 107, 312, 584, 688, 754, 756, 773-775
NEW.MEU, 551
NSLFUNC.EXE, 490, 796
opening from DOS Shell, 554-555
PRINT.COM, 555
PRINTER.SYS, 106
printing
from DOS Shell, 555-556
list, 731-737
RAMDISK.SYS, 106, 592
RAMDRIVE.SYS, 633
read-only, 243-244, 246-247, 249-251
recovering damaged, 740-743
renaming, 743-744
DOS Shell, 559
replacing on disk, 744-751
searching directories for, 435, 502, 729-731
selecting all from DOS Shell, 561
SHARE.EXE, 490, 597, 757, 765, 796
SHELL.ASC, 557, 573
SHELL.CLR, 571

SHELL.MEU, 551, 571
SHIFTIT.BAT, 440-441
source, 215, 653-654
  COPY command, 274-275
storing
  DOS Shell information, 572
  on disk, 27-28
system, 132-133
  write-protect, 250-251
TEST1.BAT, 408, 446
  TEST2.BAT, 412, 446
  TEST3.BAT, 446
  truncated by CHKDSK, 168
  TYPER.BAT, 437-438
  VDISK.SYS, 105-106, 633
  viewing from DOS Shell, 560
  WIN.INI, 373
  XMA2EMS.SYS, 105-106, 108
  XMAEM.SYS, 105-106, 108
Files are different sizes message, 265, 643
FILES configuration subcommands, 84, 456, 485, 585, 592, 649, 789
FILExxxx.CHK file, 638
filters, 4
  and piping, 364-370
  FIND, 115, 333, 364, 366-368, 370
  MORE, 262, 333, 364-367, 370, 439
  SORT, 333, 364, 368-370
FIND command, 79, 83, 679-681
  filter, 115, 333, 364, 367-368, 370
  switches
    /C, 366, 680-681
    /N, 366, 680-681
    /V, 366, 680-681
  syntax, 366
  upper/lower case, 92
FIND FIRST function call, 600
FIND: Access denied filename message, 680
FIND: File not found filename message, 680
FIND: Invalid number of parameters message, 680
FIND: Invalid parameter x message, 681
FIND: Read error in filename message, 681
FIND: Syntax error message, 681
Fixed Backup device is full message, 613
fixed disk drive, 32-33
floppy disk, 22, 33-34, 42, 124, 132, 136, 293
  see also disks *and* diskette
  buffers, 460
  checking status, 165-172
  comparing contents, 157-160
  copying, 148-157
  DISKCOPY, 150-151
  dismantling and rebuilding, 172-173, 178
  external DOS commands, 80
  formatting, 46
    with DOS Shell, 137-138
  labeling, 144-145
  logical drive, 101
  transferring system files, 148
  updating selected files, 160-165
  write protecting, 156
FLOPPY.BAT file, 413
flushing buffers, 459
FNT file extension, 237
Font file contents invalid message, 515, 723-724
fonts, 498
FOR batch subcommand, 78, 83, 618
FOR cannot be nested message, 621
FOR..IN..DO batch subcommand, 431-440, 620-621, 624
  DOS prompt, 437
  redirecting, 439
  wild-card characters, 437-439
  with other commands, 440
Format (DOS Shell) option, 565
formatting
  diskettes, 4, 41-47, 52-53, 133-136
    batch file, 402-404
    DOS Shell, 137-138
  hard disk, 138-141, 823-824
  RAM disk, 133
FORMAT command, 45, 47, 79-80, 93, 121, 125, 127-136, 139, 144, 150-151, 156, 178, 224-225, 308, 344, 347-348, 520, 565, 592, 597, 604, 609, 611-612, 623, 634, 663, 665, 669, 681-688, 693, 701, 758-759, 761, 770, 772, 792, 794, 796, 798, 823-824
switches, 127
  /1 switch, 681, 683
  /3, 127
  /4, 127, 681, 683, 792
  /8, 127, 681
  /AUTOTEST, 684
  /B, 147, 681, 683
  /BACKUP, 684
  /I, 127
  /F:size, 682, 798
  /H, 684
  /N, 796
  /N:nn, 127
  /N:ss, 682
  /S, 127, 132-133, 137-138, 140-141, 146, 592, 681, 823
  /SELECT, 684
  /T, 796
  /T:nn, 127
  /T:ttt, 682
  /V, 127, 131, 133, 137-138, 140-141, 681, 823
  /V:label, 682, 798
  /V:volume label, 127
  hard disk, 140-141
syntax, 126, 137-138
Format complete message, 823
Format failure message, 686
Format not supported on drive d: message, 686
FORMAT.BAT file, 438
FORMAT.COM file, 401, 413, 438

FORMAT.EXE, 401
Formatting... message, 823
FORTRAN language, 75
function calls
EXEC DOS, 600
FIND FIRST, 600
SEARCH FIRST, 600
function keys, 90
ANSI.SYS keyboard scan codes, 834
DOS Shell, 536
F1 (Help), 536
F10 (Start Programs/Main Group menu), 536
Shift-F9 (return to DOS), 536
EDLIN, 383-384
F1, 533-534, 814-815
F2, 546-550, 814
F3, 814-815
F4, 540, 814-815
F5, 466, 814
F6, 466, 814
F9, 815
F10, 529, 534, 815
F11, 533, 815
functions, national language support, 507-508

G

General failure error message, 156
General Failure error reading drive A: message, 31
General failure message,
GOTO batch subcommand, 78, 84, 411-414, 432, 621-622
GRAFTABL command, 79, 85, 433, 519, 524, 623, 688-690, 792, 796, 798
graphics
user interface, 318
printing on-screen, 690-691
GRAPHICS command, 79, 83, 333, 335-338, 370, 690-691, 792, 798
switches
/B switch, 335-336, 691
/LCD switch, 335-338, 691
/PB:name switch, 335-338
/PRINTBOX:x switch, 691
/R switch, 335-336
syntax, 335
GRAPHICS.PRO file, 337, 588-594
Group (DOS Shell)
option, 538
submenu, 535, 547-551
GW—BASIC program, 225

H

/H switch
FORMAT command, 684
/H:hds switch
DRIVPARM configuration subcommand, 647
/H:hh switch
DRIVER.SYS device driver, 475
hard disk, 32-33, 42, 62, 130
see also disks *and* disk drives
copying
data from floppy diskette, 408-410
files to, 824-828
drive, 22, 132
external DOS commands, 79
formats, 47
formatting, 138-141, 823, 824
logical drive, 101
partitioning, 122-126, 132, 138, 820-823
problems, 591
setting up, 817-828
hardware, 20-22, 25, 69-75
control, 97-109, 111-117
device driver 99-100, 107-108
disk drive, 100-101
display screen 99-100
input device, 70
keyboard, 89-92, 95
interrupts, 488
logical device, 70
output device, 70
peripheral, 70, 98
help system (DOS Shell), 532-533
hierarchical directory, 4, 58-62, 66-67, 182-186, 209, 211-213, 224-225, 229, 313, 346-350, 730
HLP file extension, 237
HOME.ASC file, 404-405
HOME.BAT file, 404-405

I

/I instructions option
prompt (DOS Shell) panels, 541
/I switch
DRIVPARM configuration subcommand, 647
FORMAT command, 127
/ID:nnn switch
KEYB command, 798
/ID:code switch
KEYB command, 350
I (insert lines) command, EDLIN program, 375-377, 384, 389, 397
IBM
PC, 2, 21-22
PC AT, 2, 21-22, 49, 91, 152-154, 304
date and time, 828
PC XT, 2, 49, 62, 124, 152-154, 304
PC XT 286, 153-154
PS/2, 2, 21-22, 70, 91, 152-154, 156
Model 80, 101, 105-106
IBMBIO.COM (PC DOS) file, 102, 107, 116
IBMCACHE.SYS device driver, 462

file, 106
installing, 464, 466-468
switches,
/E, 468
/NE, 468
/Psectors, 468
IBMDOS.COM file, 23, 107
IDX file extension, 237
IF subcommand, 78, 84, 432-435, 622-623
comparing strings, 433-435
looking for files, 435
testing ERRORLEVEL, 433
illegal characters, 233-235, 238, 239
Illegal Device Named message, 724
IMG file extension, 237
Incompatible system size message, 147, 774
Incompatible disks message, 662-663
Incorrect APPEND Version message, 602
Incorrect DOS version message, 147, 309, 586-587
Incorrect number of parameters message, 675, 695, 761, 772
Incorrect parameter message, 765
Index (DOS Shell) menu, 533
Infinite retry not supported on Network printer message, 724
Infinite retry on parallel printer timeout No retry on parallel printer timeout message, 724
information messages, 20, 580
initializing disks, 128, 130-141, 681-688
input/output device management, 80, 83
INPUT.COM program, 433
Ins key, 89, 92, 384, 814
Insert backup diskette nn in drive d: Strike any key when ready message, 613-614, 754
Insert disk with batch file and strike any key when ready message, 587
Insert disk with \COMMAND.COM in drive d and strike any key when ready message, 587
Insert diskette for drive d and strike any key when ready message, 587
Insert DOS disk in drive d: and strike any key when ready message, 686
Insert last backup diskette in drive d: Strike any key when ready message, 614
Insert new diskette for drive d: and strike ENTER when ready message, 686
Insert restore target in drive d: Strike any key when ready message, 754
INSTALL directive, 84, 456, 490-491, 649, 796
installing
device driver, 102,104-106, 417-418, 461-463, 468
disk, 17, 565-566
program, 17
installable device driver, 102, 104-106
installing device drivers, 417-418, 461-463, 468
Insufficient disk space - filename: message, 750
Insufficient disk space message, 587-588, 787
Insufficient memory for system transfer message, 687
Insufficient memory message, 588, 675
Insufficient memory Processing cannot continue message, 637
Insufficient room in root directory Erase files from root and repeat CHKDSK message, 637
Intel 8086 microprocessors 13, 21-22, 25, 71
memory-addressing, 108-109, 111
interface
DOS, 303-332
graphical user, 318
Intermediate file error during pipe message, 588
internal
commands, 24, 77-81, 122, 205
structure of the file system, 30
Internal error message, 678
Internal stack overflow System halted message, 588
international
characters, 106, 480-484, 645, 727-729, 759
displaying, 480-482
printing, 480, 482-484, 504-507
configuration, 106
directives, 500-507
features of DOS, 497-525
interrupt, 18, 74
hardware, 488
Invalid argument message, 678
Invalid baud rate specified message, 724
Invalid characters in volume label message, 687, 702
Invalid code page specified message, 699
Invalid COMMAND.COM in drive d message, 589
Invalid COMMAND.COM, system halted message, 589
Invalid country code message, 761
Invalid COUNTRY code or code page message, 589
Invalid date message, 306, 614, 658, 787
Invalid device message, 656
Invalid device parameters from the device driver message, 687
Invalid directory message, 192, 589, 629
Invalid disk change message, 590
Invalid drive in search path message, 195, 590, 730-731
Invalid drive or file name message, 742
Invalid drive specification message, 590, 614, 643, 671, 675, 702, 736, 750, 755, 761-762, 774, 777, 781, 787
Invalid drive specification Specified drive does not exist, or is non-removable message, 663, 667-668
Invalid environment size specified message, 590-591
Invalid filename or file not found message, 779
Invalid keyboard code message, 761

Invalid media or track ▯ bad disk unusable message, 130, 135, 687
Invalid number of parameters message, 235, 407, 591, 607, 671, 678, 742, 744, 755, 787
Invalid parameter -string- message, 724
Invalid parameter Do not give filename(s) Command format: DISKCOMP d: d: [/1] [/8] message, 663
Invalid parameter Do not specify filenames Command format: DISKCOPY d: d: [/1] message, 668
Invalid parameter message, 604, 675, 687, 691, 695, 724, 736, 750, 755, 762, 769, 772, 774, 778, 788
Invalid parameters message, 724
Invalid partition table message, 591
Invalid path, not directory, or directory not empty message, 222
Invalid path message, 591, 602, 614, 755, 762, 772, 777, 788
Invalid path or file name message, 655
Invalid path or file not found message, 607
Invalid path or parameter message, 602
Invalid path, not directory or directory not empty message, 757-758
Invalid signature in COUNTRY.SYS file message, 762
Invalid signature in KEYBOARD.SYS file message, 762
Invalid specification Specified drive does not exist, or is nonremovable message, 590
Invalid STACK parameter message, 592
Invalid sub-directory entry message, 637
Invalid switch character message, 592
Invalid syntax message, 699
Invalid time message, 614, 776
Invalid Volume ID Format failure message, 140
IO.SYS file, 102, 104, 107, 116, 312, 584, 688, 754, 756, 773-775
ISIBM.BAT file, 434-435

J

JOIN command, 79, 83, 159, 221, 333, 344, 370, 590, 608, 614, 630, 632, 659, 660, 663-664, 667, 683, 691-695, 702, 706, 730, 741, 753, 770-772, 793
  switches
    /D, 345-346, 692
  syntax, 345-347
join
  files in binary mode, 285-286
  two disk drives, 345-347

K

kernel (DOS), 23, 25, 72, 98-102, 104-108, 201, 204
  extended memory, 111
  redirecting input/output devices, 114
  service routines, 72
KEY file extension, 237
KEYB command, 79, 85, 333, 351-352, 370, 433, 498, 507, 695-700, 721, 760, 796, 798
  switches
    /ID:code, 350
    /ID:nnn, 798
  syntax, 350-351, 353
  version 3.2 and earlier, 509-510
  version 3.3 and 4.0, 510-512
KEYB has not been installed message, 699
KEYB.COM file, 490, 509-512, 524, 649, 796
keyboards, 4, 22, 89, 90-92, 95
  ANSI.SYS scan codes, 834-835
  changing layout, 695-700
  code page, 351-352
  commands (ANSI.SYS), 425-429
  country, 351-352
  enhanced, 488-489
    enable/disable extended keys, 833
  extended, 490
  key reassignment, 833
  keyboard ID, 351-352
  keyboard layout, 351-352
  selections, 351-352
  repeating character rate, 718-719
  upper/lower case, 92
KEYBOARD.SYS
  device driver, 484, 511-512, 696, 698, 759, 761-762
  selecting language, 509
KEYBxx command, 792
KEYN command, 623
keys
  ←, 813-814
  →, 92, 383, 813-814
  @, 403, 411
  Alt, 90
  Alt-F1, 815
  Backspace, 89, 92
  Break, 90
  Caps Lock, 90-91
  colon (:), 410-412
  comma (,), 387, 389-393, 394
  control, 813
  Ctrl, 90-91
  Ctrl-Alt-Del, 91, 107, 813, 818, 824
  Ctrl-Alt-F1, 512
  Ctrl-Alt-F2, 512
  Ctrl-Break, 91, 93, 101, 156-157, 261-262, 358, 365, 376-378, 403, 456-457, 616, 623, 625, 643, 813
  Ctrl-C, 91, 93, 101, 134, 262, 301, 376-378, 389, 403, 441, 456-457, 623, 643, 813
  Ctrl-I, 768
  Ctrl-Num Lock, 91, 262, 813
  Ctrl-P, 813
  Ctrl-PrtSc, 91, 444, 778, 813
  Ctrl-S, 262, 391, 444, 813

Ctrl-V, 383
Ctrl-Z, 396
Cursor, 89
Del, 89, 92, 384, 814
DOS Shell
  Help feature, 815
  special, 814-815
editing, 813-814
EDLIN, 383-384
Enter, 89, 101, 134, 376-379, 382, 386-387, 389-392, 394-395, 535, 813-814
Esc, 90-92, 382, 384, 535, 814-815
F1, 90, 92, 383
F2, 92, 383
F3, 90, 92, 378, 383
F4, 92, 384
F5, 92, 384
F6, 92, 242, 384
F7, 384
F10, 90
function, 90
  see also function keys
Ins, 89, 92, 384, 814
minus (-), 390
Num Lock, 90
Numeric keypad, 90
Pause, 90, 391
period (.), 378, 391
  PgDn, 531, 814
  PgUp, 531, 814
  plus (+), 390
  pound (#), 390
  PrtSc, 90-91 335, 337, 711, 813
  Return, 813
  Scroll Lock, 90
  Shift, 89, 90
  Shift-F9, 529, 815
  space bar, 815
  special function, 89-91
  tab, 92, 549, 815
  tilde (~), 91
kilobyte, 43
KOALA.SYS file, 106

L

/L switch
  BACKUP command, 291, 608-610, 613, 615, 795
  FC command, 269-270, 676
  RESTORE command, 298, 796
/L:numlocks switch
  SHARE command, 764-765
/L:time switch
  RESTORE command, 752-753
/LB x switch
  FC command, 676
/LBn switch
  FC command, 269-270
/LCD switch
  GRAPHICS command, 335-338, 691
/LF (DOS Shell) option, 574
L (list lines) command, EDLIN program, 377-378, 382, 385, 388, 390-393, 397
LABEL command, 79, 81, 121, 140-141, 178, 344, 347-348, 700-701, 770-772, 792-793
  syntax, 142-144
`Label not found` message, 622
labels in batch files, 410-412
`Last backup diskette not inserted` message, 614
LASTDRIVE command, 84, 456, 485-486, 604, 650, 770-771, 791
LCD (Liquid Crystal Display) video adapter, 499
LET file extension, 237
LIB file extension, 237, 269
lines
  erasing, 830
  per inch, 355
LINK command, 2, 585
list lines with EDLIN, 377-378, 385, 388, 390-393, 397
`List output is not assigned to a device` message, 736
listing
  directories, 208-209, 211-215, 658-659
  subdirectories, 658-659
`Lock violation` message, 596, 788
locking
  files, 764-766
  records, 764-766
logged disk, 88
`Logging to file filename` message, 615
logical
  device, 4, 70, 272
  disk drive, 64-67, 123, 125, 132, 474, 486
  sector, 52-53
look-ahead buffers, 461
Lotus 1-2-3, 6, 227
LPT1 (device name), 103, 340, 342
LPT2 (device name), 340
LPT3 (device name), 340
LPTn (device name), 235
`LPTx: not rerouted` message, 725
`LPTx: rerouted to COMy:` message, 725
LST file extension, 237
LTPn (device name), 103
`LTPx: set for 132` message, 725
`LTPx: set for 80` message, 725

M

/M switch
  BACKUP command, 291, 294, 296, 606-607, 611, 785
  RESTORE command, 297, 601, 752-754
  XCOPY command, 215, 217-218, 606, 782-785
/M:maxtick switch
  PRINT command, 339-343, 731

/MAINT (DOS Shell) option, 572
/MENU (DOS Shell) option, 571
/MEU (DOS Shell) option, 571
/MOS: (DOS Shell) option, 570
/MSG switch
  COMMAND command, 327
  COMMAND.COM file, 487
/MUL (DOS Shell) option, 571
M (move lines) command, EDLIN program, 385, 392, 397
MAC file extension, 237
machine code, 71, 74-75
Main Group (DOS Shell) menu, 529, 552, 570
  customizing, 549-551
maintaining disks, 121-179
MAP (Linker map file) extension, 237
mapping, bad sector, 141
MARK.BAT file, 404-405
MDA (Monochrome Display Adapter), 22, 99, 323, 499, 529
megabyte, 43
MEM command, 79, 82, 303, 309, 311-312, 331, 461, 702-705, 797
  switches
    /DEBUG, 310, 313, 703-705
    /PROGRAM, 310-312, 703-705
  syntax, 310
memory, 107
  addressing capabilities, 108-109, 111-113
  allocating, 491-493
  conventional, 109, 111-113
    page frame, 111-113
  display used and unused, 702-705
  expanded, 106, 109, 111-113, 117
    buffers, 461
    page, 111-113
  extended, 106, 109, 111, 117
  keep directory information in, 673-675
  parity error, 175
  RAM disks in 640K, 472-473
  setting number of buffers, 644-645
  storing information during hardware interrupt, 652
Memory allocation error Cannot load COMMAND, system halted message, 592
memory parity error, 175
memory-addressing capabilities, 108-109, 111-113
menu tree (DOS Shell), 527-529
MENU.BAT file, 449-450
menus
  see also submenus
  building, 449-450
  DOS Shell
    Exit, 536
    File System, 552-564, 570-571
    File System Options, 561-563
    Group, 535
    Index, 533
    Main Group, 529, 552-564
      customizing, 549-551
    Program, 535
    pull-down, 534-535
    Start Programs, 527, 529, 534-536
  enhancing up with ANSI.SYS, 450-451
  Main Group, 570
messages
  *** Files were backed up mm/dd/yy ***, 754
  *** Last file not backed up ***, 615
  *** Not able to restore file ***, 755
  *** Restoring files from drive d: ***, 755
  ***Backing up files to target drive; d: *** diskette Number: nn, 612
  . or .. Entry has a bad attribute or Entry has a bad size or Entry has a bad link, 636
  1 file(s) copied, 827
  10 Mismatches - ending compare, 643
  Abort edit (Y/N)?, 393
  Access denied, 246, 606, 671, 735, 786
  Access denied filename, 748-749
  Active code page for device devicename is codepage, 720
  Active code page not available from CON device, 698
  Add filename? (Y/N) Adding filename, 749
  All files canceled by operator, 735
  All specified file(s) are contiguous, 633
  Allocation error, size adjusted, 168
  APPEND already installed, 602
  APPEND/ASSIGN Conflict, 602
  APPEND/TopView Conflict, 602
  Attempted write-protect violation, 685
  Bad call format, 596
  Bad command, 596
  Bad command or file name, 199, 582, 779
  Bad format call, 596
  Bad or missing Command Interpreter, 582
  Bad or missing filename, 582-583
  Bad or missing Keyboard Definition File, 698
  Bad parameter, 661
  Bad Partition Table, 685
  Bad unit, 596
  Batch file missing, 583
  Cannot CHDIR to directory Tree past this point not processed, 633
  Cannot CHDIR to root Processing cannot continue, 633
  Cannot CHKDSK a Network drive, 633
  Cannot CHKDSK a SUBSTed or ASSIGNed drive, 633
  Cannot DISKCOMP to or from network drive, 661
  Cannot DISKCOPY to or from a network drive, 667
  Cannot do binary reads from a device, 655

Cannot execute A: program_name, 761
Cannot execute FORMAT, 612
Cannot format an ASSIGNed or SUBSTituted drive, 685
Cannot FORMAT nonremovable drive d:, 612-613
Cannot JOIN a network drive, 695
Cannot LABEL a network drive, 702
Cannot load COMMAND, system halted, 583
Cannot perform a cyclic copy, 786
Cannot RECOVER a Network drive, 742
Cannot start COMMAND, exiting, 583-584
Cannot SUBST a network drive, 772
Cannot SYS to a Network drive, 774
Cannot use FASTOPEN for drive d:, 674
Cannot use PRINT - Use NET PRINT, 735
Cannot XCOPY from a reserved device, 786
Cannot XCOPY to a reserved device, 786
CHDIR .. failed trying alternate method, 634
Code page nnn not prepared for all devices, 626
Code page nnn not prepared for system, 626
Code page requested (codepage) is not valid for given keyboard code, 698
Code page specified has not been prepared, 698-699
Code page specified is inconsistent with the selected code page, 698
Code pages cannot be prepared, 721
Codepage not prepared, 720
Codepage operation not supported on this device, 720-721
Compare error at offset xxxxxxxx, 642
Compare error on side xx track yy, 160
Compare error(s) on Track tt, side s, 662
Compare process ended, 662
Comparing x sectors per track, s side(s), 662
COMy bbbb,p,d,s,t, initialized, 721
Configuration too large, 584
Content of destination lost before copy, 655
Convert lost chains to files (Y/N), 168, 170-172, 334
Current CON code page: codepage, 699
Current drive is no longer valid, 584
Current keyboard code: keycode code page: codepage, 699
Current keyboard does not support this Codepage, 721
d:path\filename.ext - File not found, 642
d:path\filename.ext - File sharing conflict, 642
d:path\filename.ext - Invalid path, 643
data, 596
Delete current volume label (Y/N)?, 702
device error, 581, 595-598
Device error during Prepare, 722
Device error during Refresh, 722
Device error during Select, 722
Device error during Status, 722
Device error during write of font file to device, 722
Device or codepage missing from font file, 722
Directory Count Overflow, 777
Directory not empty, 695
directoryname Cannot recover direntry entry, processing continued, 634
directoryname Convert directory to file (Y/N)?, 635
directoryname Invalid current directory, 637
directoryname is joined, tree past this point not processed, 635
Disk boot failure, 584
Disk error reading FAT x, 635
Disk-error-type error on file filename, 737
Disk error writing FAT x, 635
Disk full error writing to BACKUP Log File Strike any key to continue, 613
Disk unsuitable for system disk, 685
Diskettes compare OK, 662
disknumber diskette bad or incompatible, 662
displaying, 410
Divide overflow, 584-585
Do you see the leftmost □? (Y/N), 324, 723
Do you see the rightmost ٩? (Y/N), 723
Does %s specify a file name or directory name on the target (F=file, D=directory)?, 786
Does WORDS specify a filename of directory name on the target (F=file D=directory)?, 216
Drive d: not ready Make sure a diskette is inserted into the drive and the door is closed, 663, 667
Drive letter must be specified, 685
Drive not ready, 596
Drive or diskette types not compatible, 662-663
Drive types or diskette types not compatible, 667
Duplicate file name or File not found, 256, 744
ECHO is status ON/OFF, 620
End of input file, 382, 386-387
Enter current Volume Label for drive d:, 686
EOF mark not found, 642
error, 580
Error during read of font file, 723
Error in COUNTRY command, 585

Error in EXE file, 585
Error loading operating system, 585
Error opening logfile, 613
Error reading partition table, 686
Error writing partition table, 686
Errors found, F parameter not specified
Corrections will not be written to the disk, 635-636
Errors on list device indicate that it may be off-line. Please check it., 735
EXEC failure, 585-586
Exiting after a fatal error to DOS, 17
Failure to access Codepage Font File, 723
Failure to access COUNTRY.SYS, 761
Failure to access device: devicename, 723
Failure to access KEYBOARD.SYS, 761
FASTOPEN already installed, 675
FASTOPEN installed, 675
fc: cannot open filename - No such file or directory, 678
fc: filename longer than filename, 678
fc: incompatible switches, 678
fc: no differences encountered, 678
fc: out of memory, 678
FCB unavailable, 596
File allocation table bad drive d:, 586, 636
File cannot be copied onto itself, 655, 749, 786-787
File creation error, 586, 761, 787
File filename canceled by operator, 736
File not found, 178, 439, 586, 626, 659, 671, 728, 742
filename Allocation error for file, size adjusted, 633
filename Contains invalid cluster, file truncated, 634
filename contains n non-contiguous blocks, 633
filename Contains xxx noncontiguous blocks, 634
filename device driver cannot be initialized, 586
filename File creation error, 754
filename File not found, 736, 787
filename File not in PRINT queue, 736
filename First cluster number is invalid, entry truncated, 636
filename has invalid cluster, file truncated, 636
filename is cross-linked on cluster X, 168
filename is currently being printed filename is in queue, 736
filename Pathname too long, 736-737
filename1 Is cross linked on cluster x filename2 Is cross linked on cluster x, 636-637
Files are different sizes, 265, 643
FIND: Access denied filename, 680
FIND: File not found filename, 680
FIND: Invalid number of parameters, 680
FIND: Invalid parameter x, 681
FIND: Read error in filename, 681
FIND: Syntax error, 681
Fixed Backup device is full, 613
Font file contents invalid, 515, 723-724
FOR cannot be nested, 621
Format complete, 823
Format failure, 686
Format not supported on drive d:, 686
Formatting..., 823
general, 581-595
General failure, 586
General Failure error, 156
General Failure error reading drive A:, 31
Illegal Device Named, 724
Incompatible disks, 662-663
Incompatible system size, 147, 774
Incorrect APPEND Version, 602
Incorrect DOS version, 147, 309, 586-587
Incorrect number of parameters, 675, 695, 761, 772
Incorrect parameter, 765
Infinite retry not supported on Network printer, 724
Infinite retry on parallel printer timeout No retry on parallel printer timeout, 724
information, 20, 580
Insert backup diskette nn in drive d:, 613-614, 754
Insert disk with batch file and strike any key when ready, 587
Insert disk with \COMMAND.COM in drive d and strike any key when ready, 587
Insert diskette for drive d and strike any key when ready, 587
Insert DOS disk in drive d: and strike any key when ready, 686
Insert last backup diskette in drive d: Strike any key when ready, 614
Insert new diskette for drive d: and strike ENTER whenready, 686
Insert restore target in drive d: Strike any key when ready, 754
Insufficient disk space, 587-588, 750, 787
Insufficient memory, 588, 675
Insufficient memory for system transfer, 687
Insufficient memory Processing cannot continue, 637
Insufficient room in root directory Erase files from root and repeat CHKDSK, 637
Intermediate file error during pipe, 588
Internal error, 678
Internal stack overflow System halted, 588
Invalid argument, 678
Invalid baud rate specified, 724

Invalid characters in volume label, 687, 702
Invalid code page specified, 699
Invalid COMMAND.COM in drive d, 589
Invalid COMMAND.COM, system halted, 589
Invalid country code, 761
Invalid COUNTRY code or code page, 589
Invalid date, 306, 614, 658, 787
Invalid device, 656
Invalid device parameters from the device driver, 687
invalid directory, 192, 589, 629
Invalid disk change, 590
Invalid drive in search path, 195, 590, 730, 731
Invalid drive or file name, 742
Invalid drive specification, 590, 614, 643, 671, 675, 702, 736, 750, 755, 761-762, 774, 777, 781, 787
Invalid drive specification Specified drive does not exist, or is non-removable, 663, 667-668
Invalid environment size specified, 590-591
Invalid filename or file not found, 779
Invalid keyboard code, 761
Invalid media or track ▯ bad-disk unusable, 130, 135, 687
Invalid number of parameters, 235, 407, 591, 607, 671, 678, 724,742, 744, 755, 787
Invalid parameter, 604, 675, 687, 691, 695, 736, 750, 755, 762, 769, 772, 774, 778, 788
Invalid parameter -string-, 724
Invalid parameter Do not give filename(s) Command format: DISKCOMP d: d: [/1] [/8] , 663
Invalid parameter Do not specify filenames Command format: DISKCOPY d: d: [/1], 668
Invalid parameter Incorrect parameter, 591
Invalid partition table, 591
Invalid path, 591, 602, 614, 755, 762, 772, 777, 788
Invalid path or file name, 655
Invalid path or file not found, 607
Invalid path or parameter, 602
Invalid path, not directory or directory not empty, 222, 757-758
Invalid signature in COUNTRY.SYS file, 762
Invalid signature in KEYBOARD.SYS file, 762
Invalid specification Specified drive does not exist, or is non-removable, 590
Invalid STACK parameter, 592
Invalid sub-directory entry, 637
Invalid switch character, 592
Invalid syntax, 699
Invalid time, 614, 776
Invalid Volume ID Format failure, 140
KEYB has not been installed, 699
Label not found, 622
Last backup diskette not inserted, 614
List output is not assigned to a device, 736
Lock violation, 596, 788
Logging to file filename, 615
LPTx: not rerouted, 725
LPTx: rerouted to COMy:, 725
LPTx: set for 132, 725
LPTx: set for 80, 725
Memory allocation error Cannot load COMMAND, system halted, 592
Missing file name, 744
Missing operating system, 592
MODE action Codepage function completed, 725
MS-DOS Version X.XX, 779
Must specify COM1, COM2, COM3 or COM4, 725
Must specify ON or OFF, 625, 780
NLSFUNC already installed, 729
NLSFUNC not installed, 322, 626-627
nnn file(s) added, 749
nnn File(s) copied, 787
nnn file(s) replaced, 749
No Append, 602
No codepage has been SELECTED, 725
No files added, 749-750
No files found filename, 750
No files replaced, 749-750
No free file handles Cannot start COMMAND, exiting, 592
No paper, 596
No Path, 198
No room for system on destination disk, 147, 774
No room in root directory, 702
No source drive specified, 615
No space left on device, 678
No sub-directories exist, 778
No such file or directory, 679
No system on default disk drive Insert system disk in drive d: and strike any key when ready, 775
No target drive specified, 615
Non-DOS disk, 597
Non-System disk or disk error Replace and strike any key when ready, 593
Not enough memory, 593
Not found, 394-395
Not ready, 597
O.K.?, 393-395
One or more CON code pages invalid for given keyboard code, 699
Out of environment space, 593
Parameter not compatible with fixed disk, 688
Parameters not compatible, 162, 688, 750
Path not found, 593, 751, 788
Path too long, 593, 788
Press any key to begin adding file(s), 751

Press any key to begin copying file(s), 788
Press any key to begin recovery of the file(s) on drive d:, 742
Press any key to begin replacing file(s), 751
Press any key to continue..., 450
Press F10 to return to DOS, 14
PRINT queue is empty, 737
PRINT queue is full, 737
Printer error, 725
Printer lines per inch set, 725
Probable non-DOS disk Continue (Y/N)?, 637
Processing cannot continue, 637-638
Program too big to fit in memory, 594
Read error on drive d:, 668
Read error, COUNTRY.SYS, 762
Read error, KEYBOARD.SYS, 762
Read fault, 597
Reading source file(s)..., 788
Replace filename? (Y/N), 751
Replacing filename, 751
Required parameter missing, 671
Resident part of PRINT installed, 737
Resident portion of MODE loaded, 725
Resynch failed. Files are too different, 679
Returning to DOS, 17
Same drive specified more than once, 675
Sector not found, 597
Sector size too large in file filename, 594
Seek, 597
SELECT is used to install DOS the first time. SELECT erases everything on the specified target and then installs DOS. Do you want to continue (Y/N)?, 762-763
SHARE already loaded, 766
Sharing violation, 594, 597, 788
SORT: Insufficient disk space, 769
SORT: Insufficient memory, 769
Source and target drives are the same, 615, 756
SOURCE diskette bad or incompatible, 668
Source does not contain backup files, 755
Source path required, 751
Strike a key when ready, 403, 414
Syntax error, 414, 434, 437, 594, 607, 623
System files restored Target disk may not be bootable, 756
System transferred, 134, 141, 688, 775, 823
Target cannot be used for backup, 615
TARGET diskette bad or incompatible, 668
Target diskette is write-protected Press any key when ready, 668
Target is full The last file was not restored, 756
Temporarily invoking DOS, 17
Too many block devices, 594
Too many drive entries, 675
Too many name entries, 675
Too many open files, 19, 679, 789
Top level process aborted, cannot continue, 595
Tree past this point not processed, 638
Unable to create directory, 595, 763, 789
Unable to create KEYB table in resident memory, 699-700
Unable to shift screen left, 726
Unable to shift screen right, 726
Unable to write BOOT, 688
Unrecognized command in CONFIG.SYS, 595
Unrecoverable file sharing error, 757
Unrecoverable read error on drive A:, 157
Unrecoverable read error on drive d: Side s, track tt Target diskette may be unusable, 669
Unrecoverable read error on drive x Track tt, side s, 663
Unrecoverable write error on drive A:, 157
Unrecoverable write error on drive B:, 157
Unrecoverable write error on drive d: Side s, track tt Target diskette is unusable, 669
usage: fc [/a] [/b] [/l] [/lbNN] [/w] [/t] [/n] [/NNNN] file1 file2, 679
Volume in drive d: has no label, 702, 781
Volume in drive d: is label_name, 702
Volume label (11 characters, ENTER for none)?, 134, 141, 688, 823
warning, 580
Warning - directory full nnn file(s) recovered, 742-743
WARNING! ALL DATA ON NONREMOVABLE DISK DRIVE d: WILL BE LOST Proceed with Format (Y/N)?, 688
Warning! Diskette is out of sequence Replace the diskette or continue if okay Strike any key when ready, 756
Warning! File filename was changed after it was backed up or is a read-only file Replace the file (Y/N)?, 756
Warning! Files in the target driver d:\ root directory will be erased, 615
Warning! No files were found to backup, 616
Warning! No files were found to restore, 756
WARNING, ALL DATA ON NON-REMOVABLE DISK DRIVE C: WILL BE LOST! Proceed with Format (Y/N)?, 823
Warning--directory full, 178
Warning: Data in the DOS partition will be lost., 126
Write error on drive d:, 668
Write fault, 597
Write protect, 597-598
x lost clusters in Y chains, 168
xxx lost clusters found in yyy chains Convert lost chains to files (Y/N)?, 638
xxxx of yyyy bytes recovered, 743

xxxxxxxxxx bytes disk space freed, 638
meta-string, 738-739
MEU file extension, 237, 548, 550
microfloppy disk, 33-34
  see also floppy disks *and* disks
microprocessors, 70, 72, 108-109,111
  Intel 8086 family, 13, 21-22, 25, 71
Microsoft QuickBASIC, 234
Microsoft Word, 373
minifloppy disk, 33-34
minus (−) key, 390
Missing file name message, 744
Missing operating system message, 592
MKDIR (MD) command, 59, 78, 81, 181, 186-187 189, 190, 229, 344, 347-348, 667, 705-706, 770-771, 825
  syntax, 188
MODE/STATUS command, 517-518
MODE action Codepage function completed message, 725
MODE CODEPAGE /STATUS command, 513, 717-718
MODE CODEPAGE PREPARE command, 509-517, 523, 626, 698-699, 712-715
MODE CODEPAGE REFRESH command, 513, 517, 716
MODE CODEPAGE SELECT command, 512-513, 516-517, 698, 714-715, 718
MODE COM PORT command, 710-712
MODE command, 79, 82, 303-304, 322, 332-333, 356, 358, 361-362, 370, 498, 507, 512-513, 626, 719-726, 796, 798
  /STATUS switch, 719
  syntax, 323-324, 326, 354-355, 357, 359-360
MODE CON command, 326
  syntax, 325
MODE DISPLAY command, 708-709
MODE KEY REPEAT command, 718-719
MODE LPTPORT command, 706-708, 711-712
MODE program, 593
MODE REDIRECTION command, 711
modem, 22
modes
  ANSI terminal codes
  reset screen mode, 832
  set graphics rendition, 831
  set screen mode, 832
  ASCII, 272-274, 276, 285, 286, 555
  binary, 272-274, 276
  display, 322-323
    real, 109, 111
  printers, 322-323
  resident, 570
  screen, 322-323
  text, 574
  transient, 570
MODUAE command, 512
Monochrome Display Adapter (MDA) 22, 99, 323, 499, 529
  ANSI.SYS screen commands, 423
MORE command, 79, 83, 726-727
  filter, 262, 333, 364, 366-367, 370, 439
  syntax, 365
MORER.BAT file, 439
MOS file extension, 237
mouse (DOS Shell), 22
  menu selection, 534
  scrolling text, 531
  COM2 port, 574
  left button, 574
mouse driver, specifying, 570
MOUSE.SYS file, 106
Move... option, 557
MOVE.BAT file, 441
MOVER.BAT file, 440
MS-DOS, *see* DOS
MS-DOS Version X.XX message, 779
MSDOS.SYS file, 23, 107, 312, 584, 688, 754, 756, 773-775
MSG file extension, 237
Multiple File List (DOS Shell) option, 563
Must specify COM1, COM2, COM3 or COM4 message, 725
Must specify ON or OFF message, 625, 780

N

/N switch
  DRIVER.SYS device driver, 476
  DRIVPARM configuration subcommand, 647
  FC command, 269-271, 676
  FIND command, 366, 680-681
  FORMAT command, 796
  RESTORE command, 297, 601, 752-754, 796
/N:nn switch
  FORMAT command, 127
/N:ss switch
  FORMAT command, 682
/NE switch
  IBMCACHE.SYS device driver, 468
/nnnn switch
  FC command, 269-270
names
  default, 227
  device, 235
  disk 141-144
  drives 478-479
  directory, 189-190
  files, 216
    changing, 252-256
    batch 401
  path, 216
NDX file extension, 237
NET PRINT command, 733
NEW.MEU file, 551
NLSFUNC already installed message, 729
NLSFUNC (national language support function)
  command, 79, 85, 498, 507-509, 626-627, 727-729, 795

program, 649
NLSFUNC not installed message, 322, 626-627
NLSFUNC.EXE file, 490, 796
nnn file(s) added message, 749
nnn File(s) copied message, 787
nnn file(s) replaced message, 749
No Append message, 602
No codepage has been SELECTED message, 725
No files added message, 749-750
No files found filename message, 750
No files replaced message, 749-750
No free file handles Cannot start COMMAND, exiting message, 592
No paper message, 596
No Path message, 198
No room for system on destination disk message, 147, 774
No room in root directory message, 702
No source drive specified message, 615
No space left on device message, 678
No sub-directories exist message, 778
No such file or directory message, 679
No system on default disk drive Insert system disk in drive d: and strike any key when ready message, 775
No target drive specified message, 615
Non-DOS disk message, 597
Non-System disk or disk error Replace and strike any key when ready message, 593
Norton Utilities, 177, 606
Not enough memory message, 593
Not found message, 394-395
Not ready message, 597
notation
batch files, 580-581
commands, 580
external commands, 580-581
NU utility, 177
NUL (device name), 103, 235
Num Lock key, 90
Numeric keypad, 90
NV (nonvolatile) RAM, 73

O

O.K.? message, 393-395
OBJ file extension, 237, 269
OLD file extension, 237
One or more CON code pages invalid for given keyboard code message, 699
Open File (DOS Shell) panel, 555
Open... (DOS Shell) option, 555
OS/2, 13, 232
Out of environment space message, 593
output
device 70, 89
redirecting, 403
OVL file extension, 237
OVR file extension, 237

P

/P prompt option
prompt (DOS Shell) panels, 541
/P switch
COMMAND command, 327, 639
DEL command, 797
DIR command, 313-316, 658, 659
ERASE command, 257, 670, 797
PRINT command, 340-343, 732
REPLACE command, 161-162, 745-747
RESTORE command, 297, 752-754
SHELL directive, 487
XCOPY command, 215-216, 219, 782-784
/PATH switch
APPEND command, 201-206, 599
/PB:name switch
GRAPHICS command, 335-338
/PRINTBOX:x switch
GRAPHICS command, 691
/PROGRAM switch
MEM command, 310-312, 703-705
/PROMPT (DOS Shell) option, 572
/Psectors switch
IBMCACHE.SYS device driver, 468
P (page lines) command, EDLIN program, 385, 390-391, 397
pages
code, 498-504, 508-509, 512-518
eject sequence, 347
frame, 111-113
memory, 111-113
PAK file extension, 237
panels, DOS Shell
Add Group, 548
Add Program, 538-542
Change Group, 548
Change Program, 546
Delete Program, 546
File View, 560
Open File, 555
prompt, 540-545
Prompt for Options, 557
Show Information, 562-563
parallel port, 354-356
redirect, 359-360
Parameter not compatible with fixed disk message, 688
parameters
adding programs from DOS Shell, 538-539, 542-545
ANSI.SYS device driver
/K, 797
/L, 797
/X, 797
batch files, 405-406
BUFFERS (CONFIG.SYS), 107
constructing batch file with, 408-410
defining or changing for block device, 647
destination, 86

DEVICE (CONFIG.SYS), 107
DISPLAY.SYS device driver, 481
drive, 195
FASTOPEN directive, 491-492
FILES (CONFIG.SYS), 107
incorrect in device driver, 586
invalid or incorrect, 591
moving with SHIFT subcommand, 440-441
PRINTER.SYS device driver, 483
RAMDRIVE.SYS device driver, 469-471
SHELL directive, 107, 487
source, 86
target, 86
VDISK.SYS device driver
/E, 799
/X, 799
`Parameters not compatible` message, 162, 688, 750
parameters of DOS, 19
parity settings, 357
parsing, 85-87, 234
partition table, 62-64, 67, 124,
partitioning hard disk, 4, 122-126, 820-822, 823
PAS file extension, 237
Pascal language, 25
path
current, 321-332
in command line, 185
PATH command, 76, 78-79, 81, 93, 122, 181, 195-201, 225-226, 229, 288, 292, 319, 329, 344, 347-348, 374, 590, 601-602, 612, 614, 729-731, 760, 763, 770, 792
search string, 196-199
syntax, 194
path names, 61-62, 216, 374-375
invalid, 591
substitute, 348-350
too long, 593
`Path not found` message, 593, 788
`Path not Found pathname` message, 751
path specifier, 192-193, 195-196, 222, 244, 276
destination, 273-275
source, 273-275
`Path too long` message, 593, 788
Pause key, 90, 391
PAUSE subcommand, 78, 84, 403-404, 431-432, 540, 618-623
PC LAN software, 208
PC Network software, 208
PC Paintbrush, 556
PCIBMDRV.MOS device driver, 570
PCMSDRV.MOS device driver, 570
PCMSPCRV.MOS device driver, 570
PCX file extension, 237, 556
period (.)
character, 86
key, 378, 391
peripheral device management, 333-370
PgDn key, 531, 814
PgUp key, 531, 814
physical disk drives, 474
PIF file extension, 237
piping (|), 3-4, 115-117, 364, 369-370
pixel, 99
platter, 32, 35, 43
plus (+) key, 390
plus (+) operator, 273-276, 281-283
port
parallel, 354-356
redirect, 359-360
serial, 354, 357-358
redirect, 359-360
pound (#) key, 390
`Press any key to begin adding file(s)` message, 751
`Press any key to begin copying file(s)` message, 788
`Press any key to begin recovery of the file(s) on drive d:` message, 742
`Press any key to begin replacing file(s)` message, 751
`Press any key to continue...` message, 450
`Press F1O to return to DOS` message, 14
PRINT command, 79, 83, 333, 338, 340-344, 347-348, 370, 379, 380, 731-737
switches
/B:bufsiz, 339-343, 731
/C, 340, 341-343, 732
/D:device, 339-342, 731
/M:maxtick, 339-343, 731
/P, 340-343, 732
/Q:maxfiles, 339-342, 731
/S:timeslice, 339-343, 731
/T, 340, 341, 342-343, 732
/U:busytick, 339-342, 731
syntax, 339-342
PRINT program, 588, 593-594
`PRINT queue is empty` message, 737
`PRINT queue is full` message, 737
Print... (DOS Shell) option, 555
PRINT.COM file, 555
`Printer error` message, 725
`Printer lines per inch set` message, 725
PRINTER.SYS device driver, 85. 106, 480, 482-484, 498, 500, 502, 504-507, 509, 514-515, 518, 626, 713, 718, 721-722, 794, 798
parameters, 483
printers
cancel printing, 340-341
out of paper, 597
parallel port, 355
printing graphics-screen, 690-691
queueing, 340-341
SELECT utility, 567-568
serial vs parallel, 711
setting IBM characteristics, 706-708
terminate printing, 340-341
types, 335-336
printing
background color, 336

graphics screens, 335-338
image reversed, 336
international characters, 480, 482-484, 504-507
list of files, 731-737
queue, 734
screen image (GRAPHICS), 338
serial printer instead of parallel printer, 711
PRN (device name), 103, 209, 214, 228, 235, 340, 342, 380
PRN file extension, 237
PRO file extension, 237
`Probable non-DOS disk Continue (Y/N)?` message, 637
`Processing cannot continue,` message, 637-638
ProDOS, 12
Program (DOS Shell) submenu, 537-547, 550
`Program too big to fit in memory` message, 594
programmable ROM (EPROM), 73
programming
DOS, 18
languages, 25, 74-75
programs, 72
accessing from DOS Shell, 529
adding from DOS Shell, 538-545, 548, 550
applications, 21, 78
ASK.COM, 433
assembler, 74-75
associating with file extensions, 556-557
BASIC, 353, 556
BASICA, 225
changing
from DOS Shell, 545-546, 548-549
order of listings in DOS Shell, 549
COMMAND.COM, 17, 21, 23-25, 69-70, 72-73, 77-81, 85-89, 93-95, 97-98, 107, 116, 122, 127, 132, 145, 148, 165, 189, 225, 234-235, 250-251, 255-256, 275, 292
copying from DOS Shell, 546-547, 550-551
DEBUG, 225, 635
deleting from DOS Shell, 546, 549
device driver routines, 100-101
displaying all in system, 702-705
DisplayWrite 4, 556
dividing by zero, 584-585
DUMP.EXE, 177
EDLIN, 106, 225, 373-397, 401, 418, 421, 423, 428-429
FASTOPEN, 312, 649
FDISK, 591, 817-823
GRAPHICS, 588, 593-594
GW—BASIC, 225
INPUT.COM, 433
Install, 17
KEYB, 509-512, 524, 649
MODE, 593
NLSFUNC, 649
Norton Utilities, 606
PC Paintbrush, 556
PRINT, 588, 593-594
ProKey, 588, 593
Publisher's Paintbrush, 556
QEMM-386, 460
Quattro, 556
SDACHMD, 606
SELECT, 17, 152, 165
SETUP, 307-308, 817, 828
SHARE.EXE, 594, 596, 649
SHELLB.COM, 569
SHELLC.COM, 569
SideKick, 588, 593-594
starting
from CONFIG.SYS file, 649
from DOS Shell, 537-538, 552
stopping, 625
Ctrl-C or Ctrl-Break, 643
XCOPY.EXE, 374
ProKey program, 588, 593
prompt (DOS Shell) panels, 540-545
/I instructions option, 541
/P prompt option, 541
/T title option, 541
PROMPT command, 76, 78, 82, 93, 187, 288, 300, 303-304, 319, 321, 328-329, 332, 420-423, 428-429, 737-740, 763
meta-strings
$b (Vertical bar), 420
$d (Date), 420
$e (Escape character), 420-421
$g (Greater than symbol), 420-421
$h (Backspace), 420
$l (Less than symbol), 420-421
$n (Default drive name), 420
$p (Current drive and path), 420
$q (Equal sign), 420
$t (Time), 420
$v (DOS version number), 420
$_ (Underscore [line feed]), 420
syntax, 320-321
Prompt for Options (DOS Shell) panel, 557
prompts
customizing, 737-740
DOS prompt, 17, 24, 69, 76, 93, 375, 379-380
EDLIN program
* prompt, 375-380, 382, 386-387, 389, 395-396
? prompt, 393-395
protected mode, 109, 111, 117
PrtSc (print screen) key, 90-91, 335, 337, 813
PS file extension, 237
Publisher's Paintbrush, 556

Q

/Q:maxfiles switch
PRINT command, 339-342, 731
Q (quit edit) command
EDLIN program, 379, 381-382, 385, 392-393, 397
QEMM-386 program, 460
QNX, 63-65
Quattro, 556

queue, 734
Quietwriter III 5202, 499, 505
QWERTY keyboard layout, 89

R

/R switch
  GRAPHICS command, 335-336, 691
  REPLACE command, 161-162, 745-747
  SORT command, 368, 766
R (replace text) command, EDLIN program, 385, 394-397
RAM (Random Access Memory), 23, 25, 71, 73, 76, 94, 101, 175, 458, 588, 593
  buffers, 107, 274, 276
  installing disk device, 105-106
  nonvolatile, 73
  stack, 488
RAM disks, 463-464, 469-473, 486, 826-827
  formatting, 133
  640K memory, 471-472
  extended or expanded memory, 472-473
RAMDRIVE.SYS device driver, 106, 469-471, 478-479, 592, 646
  /E, 470, 592
  /E:max, 469-471
  /X, 469, 471
  parameters, 469-471
RAMDRIVE.SYS file, 633
Random Access Memory *See* RAM
RD command, 634, 757-758
`Read error on drive d:` message, 668
`Read error, COUNTRY.SYS` message, 762
`Read error, KEYBOARD.SYS` message, 762
`Read fault` message, 597
Read Only Memory, *See ROM*
read-only file, 243-244, 246-247, 249-251
read-write attribute, 243-244, 246-247, 249-251
`Reading source file(s)...` message, 788
real mode, 109, 111
records, locking, 764-766
RECOVER command, 79, 81, 122, 172, 177-179, 586, 671, 740-743
  directory entries, 176, 189
  file, 174-175
  syntax, 173-176
recovering damaged files, 740-743
redefining scan codes, 428-429
redirection, 3, 114-115, 262-264, 362-364, 369-370
  file I/O, 344-345
  FOR..IN..DO subcommand, 439
  syntax, 262
  input (<) symbol, 114-115, 362
  input/output, 114-115, 214-215, 362-364
  output (>) symbol, 362
  output (>>) symbol, 115, 362
  parallel to serial port, 359-360
Reference disk, 464, 817
REM command, 78, 84, 410-411, 432, 457, 490, 620, 624, 650, 796
remarks
  in batch files, 410-411
  in CONFIG.SYS file, 490, 650
removing
  directories, 220-223
  subdirectories, 757-758
RENAME (REN) command, 78, 82, 231, 253-256, 302, 583, 743-744
  syntax, 251-252
Rename... (DOS Shell) option, 559
renaming files, 743-744
Reorder... (DOS Shell) option, 549
REPLACE command, 79, 81, 121, 161-162, 179, 287, 433, 623, 744-751, 794, 798
  switches
    /A, 161-164, 745-747
    /D, 161-163
    /P, 161-162, 745-747
    /R, 161-162, 745-747
    /S, 161-164, 745-748
    /U, 161-163, 745-748, 798
    /W, 161-162, 745-747
  syntax, 160-161
  update new version DOS, 163-165
`Replace filename? (Y/N)` message, 751
`Replacing filename` message, 751
`Required parameter missing` message, 671
resident device driver, 102
  character, 103-104
  reserved names, 103-104
resident mode, 570
`Resident part of PRINT installed` message, 737
`Resident portion of MODe loaded` message, 725
RESTORE command, 33, 79, 82, 139, 172, 218, 231, 244-245, 287, 289-290, 296, 298-302, 344, 347-348, 433, 565, 601, 604, 623, 634, 693, 751-757, 770, 772, 792, 796
  restore modified files switch, 245
  switches
    /A, 298, 752-753
    /B, 297, 796
    /B:date, 752-753
    /E, 298, 796
    /E:time, 752-753
    /L, 298, 796
    /L:time, 752-753
    /M, 297, 601, 752-754
    /N, 297, 601, 752-754, 796
    /P, 297, 752-754
    /S, 297, 752-753
  syntax, 297
Restore Fixed Disk (DOS Shell) option, 565
restoring backup files, 296-302, 751-757
`Resynch failed. Files are too different` message, 679
Return key, 813
`Returning to DOS` message, 17

RFT file extension, 237
RMDIR (RD) command, 59, 78, 81, 182, 229, 344, 347-348, 670, 693, 757-758, 770-771
  syntax, 220
ROM (Read Only Memory), 23, 25, 71, 73, 76, 94
  display adapter, 99
root directory, 58-63, 67, 131, 149, 168, 182-186, 209, 220-222, 224-226, 294, 329
  capacity, 130
  drive name, 184-185
  rebuild with RECOVER, 176-178
  renaming, 346-347
root file name, 232, 234, 253, 255-256, 317
routines
  bootstrap loader, 107-108
  kernel, 72, 204, 209

S

/S switch
  ATTRIB command, 243, 605, 795
  BACKUP command, 291, 296, 607, 610-611
  FORMAT command, 127, 132-133, 137-138, 140-141, 146, 592, 681, 823
  REPLACE command, 161-164, 745-748
  RESTORE command, 297, 752-753
  XCOPY command, 215-216, 217, 219, 782-784
/S:sec switch
  DRIVPARM configuration subcommand, 647
/S:ss switch
  DRIVER.SYS device driver, 475
/S:timeslice switch
  PRINT command, 339-343, 731
  SELECT switch
  FORMAT command, 684
/SND (DOS Shell) option, 571
/STATUS switch
  MODE CODEPAGE command, 717-718
  MODE command, 719
/SWAP (DOS Shell) option, 572
S (search test) command, EDLIN program, 385, 393-397
`Same drive specified more than once` message, 675
SAV file extension, 237
scan codes, 425-429
  extended, 425, 427
  redefining, 428-429
screens
  appearance, 332
  Change Disk Cache Settings, 466
  clearing, 334, 411
  columns, 325
  commands
    ANSI.SYS file, 423-424
  DOS Shell
    Add Group, 535
    Add Programs, 535
    erasing, 638-639
    lines, 325
    Specify Function and Workspace, 566
scroll bar (DOS Shell), 531-532
Scroll Lock key, 90-91
SDACHMD program, 606
SEARCH FIRST function call, 600
searching
  path, 196-199
  string, 196-199
`Sector not found` message, 597
`Sector size too large in file filename` message, 594
sector-to-cluster ratio, 129
sectors, 149
  cluster, 48-50, 128
  defective
    FAT (file allocation table), 166
    file allocation table (FAT), 166
  identification, 131
  log, 48-51
  logical, 41-43, 48-50, 52-53, 65
  track, 43, 46-47, 128, 149
`Seek` message, 597
Select across directories (DOS Shell) option, 562
Select All (DOS Shell) option, 558, 561
SELECT command, 79, 81, 758-763, 792, 794, 798
`Select error during Prepare` message, 722
`SELECT is used to install DOS the first time. SELECT erases everything on the specified target and then installs DOS. Do you want to continue (Y/N)?` message, 762-763
SELECT utility, 17, 152, 165, 519-521
  configuring, 566-567
  DOSSHELL.BAT file, 568-569
  drive and printer information, 567-568
  installing DOS version 4 and DOS Shell, 565-569
  version 3.2 and 3.3, 519-521
  version 3.x, 519
  version 4.0, 519
semicolon (;) character, 195, 196, 198, 203
serial
  device, 358
  port, 354, 357-359
  redirect, 359, 360
SET command, 78, 83, 93, 187, 196-198, 200-201, 288, 304, 329, 331-332, 446, 593, 730, 737, 763, 764
  syntax, 330
Set Date/Time (DOS Shell) options, 565
SET PATH command, 122
SETUP program, 2, 307-308, 817, 828
`SHARE already loaded` message, 766
SHARE command, 79, 84, 132, 484, 764-766, 792
  switches
    /F:name_space, 764-765
    /L:numlocks, 764-765
SHARE.EXE file, 490, 594, 596-597, 649, 757, 765, 796
`Sharing violation` message, 594, 597, 788
shell, 24, 224-225
SHELL (CONFIG.SYS file) directive, 582
Shell Change attribute... (DOS Shell) option, 559-560

SHELL directive, 84, 457, 486-488, 589-591, 593, 651, 640 793-794
switches
/E, 487
/E:size, 487, 590-591, 793-794
/P, 487
parameters, 487
SHELL.ASC file, 557, 573
SHELL.CLR file, 571
SHELL.MEU file, 551, 571
SHELLB.COM program, 569
SHELLC.COM program, 569
Shift key, 89-90
SHIFT command, 78, 84, 414, 432, 440-441, 616, 624
moving parameters, 440-441
Shift-F9 key, 529, 536, 815
Shift-PrtSc key, 91, 711
SHIFTIT.BAT file, 440-441
Show Information (DOS Shell) panel, 562-563
SideKick program, 588, 593-594
slash (/) key, 87
SMARTDRV.SYS device driver, 462, 468
/A switch, 468
installing, 468
SORT command, 79, 83, 766-769
filter, 116, 333, 364, 370
piping, 369-370
redirection, 369-370
input/output, 114, 115
standard input/output, 113
switches
/+c, 766
/R, 368, 766
syntax, 368
`SORT: Insufficient disk space` message, 769
`SORT: Insufficient memory message`, 769
sorting, ASCII, 766-769
sounds, controlling in DOS Shell, 571
source
device, 274-275
disk, 150-151, 664
file, 215, 274-275, 653-654
specifier, 86
`Source and target drives are the same` message, 615, 756
`SOURCE diskette bad or incompatible` message, 668
`Source does not contain backup files` message, 755
`Source path required` message, 751
space bar key, 815
special ASCII characters
backslash (\), 86, 184, 186, 192
colon (:), 86
Ctrl-@ (null), 384
Ctrl-G, 234
Ctrl-S, 234, 240-242
Ctrl-Z, 241-242, 261, 264, 267, 272, 275-277, 285-286, 302, 383, 384
equal (=), 197
Escape, 383
period (.), 86
piping joint (|) symbol, 115, 116, 364
plus (+) operator, 273-276, 281-283
redirection input (<) symbol, 114-115, 362
redirection output (>) symbol, 362
redirection output (>>) symbol, 115, 362
semicolon (;), 195-196, 198, 203
slash (/), 87
specifiers, 86
destination file, 86
dot dot (..), 279-280, 318
drive, 192-193, 195, 222, 244, 276, 292
destination, 273-275
source, 273-275
file, 261-262, 276, 292
path, 192-193, 195-196, 222, 244, 276, 292
destination, 273-275
source, 273-275
source file, 86
Specify Function and Workspace screen, 566
STACKS directive, 84, 457, 488-489, 588, 592, 652, 793
Start (DOS Shell) option, 537-538
Start Programs (DOS Shell) menu, 527, 529, 534-536
stopping command, 93
`Strike a key when ready` message, 414
`Strike a key when ready...` message, 403
string, 328-329
comparing with IF subcommand, 433-435
displaying from files, 679-681
environment, 94
meta-string, PROMPT command, 320
search (PATH), 196-199
STY file extension, 237
subcommands, 399
see also commands
@, 432
batch file, 431-446
CALL, 432, 442-445, 795
CLS, 432
COMMAND, 432, 442, 445-446
ECHO ON/OFF, 410, 411, 431-432
FOR..IN..DO, 431-432, 435-440
GOTO, 411-414, 432
IF, 432-435
PAUSE, 403-404, 431-432
REM, 410-411, 432
SHIFT, 414, 432, 440-441, 616
subdirectories, 58-62, 130, 168, 172, 348-350, 705-706, 825
alias disk drive name, 769-772
changing current, 191-194
comparing files, 436-437
damage to entries, 166
default names, 227
deleting, 220-223
directory entry, 183
displaying from disk, 776-778
joining with disk drive, 691-695

listing, 658-659
making, 187-190
removing, 220-223, 757-758
repairing, 176-178
root directory, 184-186
search path, 198-199
sorting, 369-370
switch, 216-218
temporary files, 228-229
submenus, DOS Shell
File, 554-561
Program, 537-547, 550
SUBST command, 79, 83, 133, 144, 159, 221, 333, 344, 346, 370, 590, 604, 608, 614, 630, 632-633, 650, 660, 663-664, 667, 683, 685, 692, 706, 730, 741, 753, 769-772, 793
/D switch, 770, 772
syntax, 347-350
SWITCHAR directive, 791
switches, 87
+A/-A
ATTRIB command, 794
/+c
SORT command, 766
/1
DISKCOMP command, 158, 660-661
DISKCOPY command, 149, 664-665
FORMAT command, 681, 683
/3
FORMAT command, 127
/4
FORMAT command, 127, 681, 683, 792
/8
DISKCOMP command, 158, 660-661
FORMAT command, 127, 681
/A
BACKUP command, 291-292, 294-296, 607, 609, 611, 613-615
COPY command, 273-277, 653-655
FC command, 269, 676
REPLACE command, 161-164, 745-747
RESTORE command, 298
SMARTDRV.SYS device driver, 468
TREE command, 209, 777
XCOPY command, 215, 217-219, 606, 782, 784
/A:date
RESTORE command, 752-753
/AUTOTEST
FORMAT command, 684
/B
COPY command, 273, 276-277, 285-286, 653-655
EDLIN program, 381
FC command, 268, 270, 676-677
FORMAT command, 147, 681, 683
GRAPHICS command, 335-336, 691
RESTORE command, 297, 796
/B:bufsiz
PRINT command, 339-343, 731
/B:date
RESTORE command, 752-753
/BACKUP
FORMAT command, 684
/C
COMMAND command, 619, 639, 445-446
DRIVER.SYS device driver, 476
DRIVPARM configuration subcommand, 647
FC command, 269, 676-677
FIND command, 366, 680-681
PRINT command, 340-343, 732
/C string
COMMAND command, 327-328
/D
BACKUP command, 291
JOIN command, 345-346, 692
REPLACE command, 161-163
SUBST command, 770, 772
/D:date
BACKUP command, 607, 614
XCOPY command, 215, 217-219, 782, 785
/D:ddd
DRIVER.SYS device driver, 474-475
/D:device
PRINT command, 339-342, 731
/DEBUG
MEM command, 310, 313, 703-705
/E
APPEND command, 200-203, 206, 599-601
IBMCACHE.SYS device driver, 468
RAMDISK.SYS file, 592
RAMDRIVE.SYS device driver, 470
RESTORE command, 298, 796
SHELL directive, 487
XCOPY command, 215-217, 782, 784
/E:eeeee
COMMAND command, 327-328
/E:max
RAMDRIVE.SYS device driver, 469-471
/E:size
COMMAND command, 639
SHELL directive, 487, 590-591, 793-794
/E:time
RESTORE command, 752-753
/F
BACKUP command, 291-293, 295, 608-609, 612-613, 682, 795
CHKDSK command, 166, 586, 629-631, 634-638
TREE command, 139, 209, 212-213, 659, 777, 793
/F:f
DRIVER.SYS device driver, 475
/F:name_space
SHARE command, 764-765
/F:size
BACKUP command, 291, 608-609
FORMAT command, 682, 798
/F:type
DRIVPARM configuration subcommand, 647

/H
FORMAT command, 684
/H:hds
DRIVPARM configuration subcommand, 647
/H:hh
DRIVER.SYS device driver, 475
/I
DRIVPARM configuration subcommand, 647
FORMAT command, 127
/ID:nnn
KEYBOARD command, 798
/ID:code
KEYB command, 350
/L
BACKUP command, 291, 608, 610, 613, 615, 795
FC command, 269-270, 676
RESTORE command, 298, 796
/L:numlocks
SHARE command, 764-765
/L:time
RESTORE command, 752-753
/LB x
FC command, 676
/LBn
FC command, 269-270
/LCD
GRAPHICS command, 335-338, 691
/M
BACKUP command, 291, 294, 296, 606-607, 611, 785
RESTORE command, 297, 601, 752-754
XCOPY command, 215, 217-218, 606, 782-785
/M:maxtick
PRINT command, 339-343, 731
/MSG
COMMAND command, 327
COMMAND.COM file, 487
/N
DRIVER.SYS device driver, 476
DRIVPARM configuration subcommand, 647
FC command, 269-271, 676
FIND command, 366, 680-681
FORMAT command, 796
RESTORE command, 297, 601, 752-754, 796
/N:nn
FORMAT command, 127
/N:ss
FORMAT command, 682
/NE
IBMCACHE.SYS device driver, 468
/nnnn
FC command, 269-270
/P
COMMAND command, 327, 639
DEL command, 797
DIR command, 313-316, 658-659
ERASE command, 257, 670, 797
PRINT command, 340-343, 732
REPLACE command, 161-162, 745-747
RESTORE command, 297, 752-754
SHELL Directive, 487
XCOPY command, 215-216, 219, 782-784
/PATH OFF/ON
APPEND command, 201, 203-204, 206, 599
/PB:name
GRAPHICS command, 335-338
/PRINTBOX:x
GRAPHICS command, 691
/PROGRAM
MEM command, 310-312, 703-705
/Psectors
IBMCACHE.SYS device driver, 468
/Q:maxfiles
PRINT command, 339-342, 731
/R
GRAPHICS command, 335-336, 691
REPLACE command, 161-162, 745-747
SORT command, 368, 766
/S
ATTRIB command, 243, 605, 795
BACKUP command, 291, 296, 607, 610-611
FORMAT command, 127, 132-133, 137-138, 140-141, 146, 592, 681, 823
REPLACE command, 161-164, 745-748
RESTORE command, 297, 752-753
XCOPY command, 215-217, 219, 782-784
/S:sec
DRIVPARM configuration subcommand, 647
/S:ss
DRIVER.SYS device driver, 475
/S:timeslice
PRINT command, 339-343, 731
/SELECT
FORMAT command, 684
/STATUS
MODE CODEPAGE command, 717-718
MODE command, 719
/T
BACKUP command, 291, 795
FC command, 270, 676
FORMAT command, 796
PRINT command, 340-343, 732
/T:nn
FORMAT command, 127
/T:time
BACKUP command, 608, 614
/T:trk
DRIVPARM configuration subcommand, 648
/T:ttt
DRIVER.SYS device driver, 475
FORMAT command, 682
/U
REPLACE command, 161-163, 745-746, 748, 798
/U:busytick
PRINT command, 339-342, 731
/V
CHKDSK command, 166, 213, 629, 631, 637, 659
COPY command, 273, 275, 287, 406, 653, 825-826

FIND command, 366, 680-681
FORMAT command, 127, 131, 133, 137-138, 140-141, 681, 823
XCOPY command, 215-216, 782
/V:label
FORMAT command, 682, 798
/V:volume label
FORMAT command, 127
/W
DIR command, 313, 316, 658-659
FC command, 270-271, 676-677
REPLACE command, 161-162, 745-747
XCOPY command, 215-216, 782, 785
/X
APPEND command, 200-204, 206-208, 599-602, 783, 785
BUFFERS configuration subcommand, 461, 644, 797
FASTOPEN command, 492, 674, 798
FC command, 676, 678
RAMDRIVE.SYS device driver, 469, 471
/X:aa
XMA2EMS.SYS device driver, 479
/X:OFF/ON
APPEND command, 599-600
SWITCHES command, 84
`Syntax error` message, 414, 434, 437, 594, 607, 623
SYS file extension, 237
SYS command, 79, 81, 121, 132, 145, 147-148, 162, 164, 179, 585,595, 754, 756, 773-775, 798
syntax, 146, 149
SYS file extension, 237, 269, 825
system
board, 71
clock, 74
configuration management, 80
commands, 84
date, 657-658
environment, setting or showing, 763-764
files, copy, 132
information management, 80
commands, 82-83, 303-332
level hardware components, 21-22, 25
programs, write protect, 250-251
time, 305-308, 775-776
timer, 305
System file list (DOS Shell) option, 564
`System files restored Target disk may not be bootable` message, 756
system level hardware components, 21-22, 25
system programs, write-protect, 250-251
system timer, 305
`System transferred` message, 134, 141, 688, 775, 823

T

/T switch
BACKUP command, 291, 795
FC command, 270, 676
FORMAT command, 796
PRINT command, 340-343, 732
/T title option
prompt (DOS Shell) panels, 541
/T:nn switch
FORMAT command, 127
/T:time switch
BACKUP command, 608, 614
/T:trk switch
DRIVPARM configuration subcommand, 648
/T:ttt switch
DRIVER.SYS device driver, 475
FORMAT command, 682
/TEXT (DOS Shell) option, 574
/TRAN (DOS Shell) option, 570
T (transfer lines) command, EDLIN program, 385, 396
tab key, 92, 549, 815
`Target cannot be used for backup` message, 615
`TARGET diskette bad or incompatible` message, 668
`Target diskette is write-protected Press any key when ready` message, 668
`Target is full The last file was not restored` message, 756
`Temporarily invoking DOS` message, 17
TEST1.BAT file, 408, 446
TEST2.BAT file, 412, 446
TEST3.BAT file, 446
testing ERRORLEVEL, 433
TIF file extension, 237
time
displaying from DOS Shell, 573
for background printing, 339-340
Personal Computer AT, 828
setting and showing for system, 775-776
TIME command, 76, 78, 83, 303, 305, 307-308, 331, 446-447, 565, 617, 657, 753, 760, 775-776, 795, 798, 828
syntax, 307
TMP file extension, 237
`Too many block devices` message, 594
`Too many drive entries` message, 675
`Too many name entries` message, 675
`Too many open files` message, 19, 679, 789
`Top level process aborted, cannot continue` message, 595
Toshiba
CHAD, 2
hard RAM disk, 133
track, 35, 37, 40, 46-48, 62, 65, 130-131
transfer
lines with EDLIN, 385, 39
system files, 145-148

transient mode, 570
TREE command, 79, 81, 141, 145, 177, 182, 208, 211-212, 214-215, 226, 229, 701, 776-778, 793, 798
  alternate graphics switch, 209
  compare with CHKDSK, 213
  file list switch, 209, 212-213
  output to printer, 139-140, 209
  switches
    /A, 209, 777
    /F, 139, 209, 212-213, 659, 777, 793
  syntax, 209
Tree past this point not processed message, 638
TRUENAME command, 797
TST file extension, 237
TXT file extension, 237, 641, 744
TYPE command, 78, 82, 169, 177-178, 231, 240-242, 261-263, 284-286, 302, 379-380, 419, 421, 437-438, 741, 769, 778-779
  output to printer, 261
  syntax, 260
TYPER.BAT file, 437-438

U

/U switch
  REPLACE command, 161-163, 745-746, 748, 798
/U:busytick switch
  PRINT command, 339-342, 731
Unable to create directory message, 595, 763, 789
Unable to create KEYB table in resident memory message, 699-700
Unable to shift screen left message, 726
Unable to shift screen right message, 726
Unable to write BOOT message, 688
UNIX, 13, 30, 58, 63, 124, 226, 232, 272
  CAT command, 272
Unrecognized command in CONFIG.SYS message, 595
Unrecoverable file sharing error message, 757
Unrecoverable read error on drive A: message, 157
Unrecoverable read error on drive d: Side s, track tt Target diskette may be unusable message, 669
Unrecoverable read error on drive x Track tt, side s message, 663
Unrecoverable write error on drive A: message, 157
Unrecoverable write error on drive B: message, 157
Unrecoverable write error on drive d: Side s, track tt Target diskette is unusable message, 669
updating selected files, 160-165
upgrading DOS with REPLACE command, 163-165
usage: fc [/a] [/b] [/l] [/lbNN] [/w] [/t] [/n] [/NNNN] file1 file2 message, 679
user interface, 318
utilities
  accessing from DOS Shell, 529
  Change Colors, 564, 570
  command program 77-79
  FASTBACK, 129
  FDISK, 63-64, 67
  loading into CONFIG.SYS file, 490-491
  NU, 177
  SELECT, 519-521

V

/V switch
  CHKDSK command, 166, 213, 629, 631, 637, 659
  COPY command, 273, 275, 287, 406, 653, 825-826
  FIND command, 366, 680-681
  FORMAT command, 127, 131, 133, 137-138, 140-141, 681, 823
  XCOPY command, 215-216, 782
/V:label switch
  FORMAT command, 682, 798
/V:volume label switch
  FORMAT command, 127
variables
  markers, batch files, 406-408
  environmental, COMSPEC, 587, 763
VDISK
  formatting, 133
  device driver, 101-102
VDISK.SYS device driver, 105-106, 469, 633, 791, 799
  parameters
    /E, 799
    /X, 799
VER command, 78, 83, 303, 308, 331, 760, 779
  syntax, 309
VERIFY command, 78, 82, 231, 286, 302, 609, 779-780
  syntax, 287
  verify BACKUP operation, 288-289
verify switch, 216
video adapters, 499
Video Graphics Adapter (VGA), 22, 70, 99, 324-326, 529-530
video graphics mode
  specifying for DOS Shell, 573-574
View (DOS Shell) option, 560
viewing
  files, 260-263
  directories, 313-318
virtual disks, 469
VOL command, 78, 81, 121, 144-145, 178, 781
Volume in drive d: has no label message, 702, 781
Volume in drive d: is label_name message, 702
volume label
  assigned, 142-143
  defined, 131-132
  disks, 700-702
  displayed, 141, 781

entered, 127
stored, 142-143
Volume label (11 characters, ENTER for none)?
message, 134, 141, 688, 823
volume serial number, 131-132, 135

W

/W switch
DIR command, 313, 316, 658-659
FC command, 270-271, 676-677
REPLACE command, 161-162, 745-747
XCOPY command, 215-216, 782, 785
W (write lines) command, EDLIN program, 385-387, 397
warm boot, 91, 196
Warning - directory full nnn file (s) recovered message, 742-743
warning messages, 580
WARNING! ALL DATA ON NONREMOVABLE DISK DRIVE d: WILL BE LOST Proceed with Format (Y/N) message, 688
Warning! Diskette is out of sequence Replace the diskette or continue if okay Strike any key when ready message, 756
Warning! File filename was changed after it was backed up or is a read-only file Replace the file (Y/N)? message, 756
Warning! Files in the target drive d:\ root directory will be erased message, 615
Warning! No files were found to backup message, 616
Warning! No files were found to restore message, 756
WARNING, ALL DATA ON NON-REMOVABLE DISK DRIVE C: WILL BE LOST! Proceed with Format (Y/N)?, 823
Warning--directory full message, 178
Warning: Data in the DOS partition will be lost. message, 126
wild card
(*), 215, 217-218, 252-256, 259-260, 264-265, 267, 270, 278-279, 282, 289, 300-302, 317
(?), 254-255
WIN.INI file, 373
Winchester disk, 32
Windows, 6
multiple from DOS Shell, 563-564
WK1 file extension, 237
WK3 file extension, 237
WKQ file extension, 237, 573
WKS file extension, 237
word processor, 373
WordPerfect, 6, 169, 325, 373
WordStar, 200-201, 373
WP file extension, 556
Write error on drive d: message, 668
Write fault message, 597
write protect
applications program files, 246
system program files, 250-251
Write protect message, 597-598
writing
files, 31
data, 40-42

X

/X switch
APPEND command, 200-208, 599-602, 783, 785
BUFFERS configuration subcommand, 461, 644, 797
FASTOPEN command, 492, 674, 798
FC command, 676, 678
RAMDRIVE.SYS device driver, 469, 471
/X:aa switch
XMA2EMS.SYS device driver, 479
/X:OFF/ON switch
APPEND command, 599-600
lost clusters in Y chains message, 168
XCOPY command, 33, 79, 81, 139, 149, 157, 182, 216-218, 229, 244, 287, 294, 308, 400, 433, 520, 540, 544, 667, 759, 761, 781-789, 794
switches
/A, 215, 217-219, 606, 782, 784
/D:date, 215, 217-219, 782, 785
/E, 215-217, 782, 784
/M, 215-218, 606, 782-785
/P, 215-216, 219, 782-784
/S, 215-217, 219, 782-784
/V, 215-216, 782
/W, 215-216, 782, 785
syntax, 215
XCOPY.EXE program, 374
XENIX, 63, 124
XMA2EMS.SYS device driver, 105-106, 108, 479, 796
/X:aa switch, 479
XMAEM.SYS device driver, 105-106, 108, 479-480, 796
xxx lost clusters found in yyy chains Convert lost chains to files (Y/N)? message, 638
xxxx of yyyy bytes recovered message, 743
xxxxxxxxxxx bytes disk space freed message, 638

Free Catalog!

Mail us this registration form today, and we'll send you a free catalog featuring Que's complete line of best-selling books.

Name of Book _____

Name _____

Title _____

Phone ( ) _____

Company _____

Address _____

City _____

State _____ ZIP _____

Please check the appropriate answers:

1. Where did you buy your Que book?
   - ☐ Bookstore (name: _____)
   - ☐ Computer store (name: _____)
   - ☐ Catalog (name: _____)
   - ☐ Direct from Que
   - ☐ Other: _____
2. How many computer books do you buy a year?
   - ☐ 1 or less
   - ☐ 2-5
   - ☐ 6-10
   - ☐ More than 10
3. How many Que books do you own?
   - ☐ 1
   - ☐ 2-5
   - ☐ 6-10
   - ☐ More than 10
4. How long have you been using this software?
   - ☐ Less than 6 months
   - ☐ 6 months to 1 year
   - ☐ 1-3 years
   - ☐ More than 3 years
5. What influenced your purchase of this Que book?
   - ☐ Personal recommendation
   - ☐ Advertisement
   - ☐ In-store display
   - ☐ Price
   - ☐ Que catalog
   - ☐ Que mailing
   - ☐ Que's reputation
   - ☐ Other: _____
6. How would you rate the overall content of the book?
   - ☐ Very good
   - ☐ Good
   - ☐ Satisfactory
   - ☐ Poor
7. What do you like *best* about this Que book?

   _____

   _____
8. What do you like *least* about this Que book?

   _____

   _____
9. Did you buy this book with your personal funds?
   - ☐ Yes ☐ No
10. Please feel free to list any other comments you may have about this Que book.

    _____

    _____

    _____

QUE

Order Your Que Books Today!

Name _____

Title _____

Company _____

City _____

State _____ ZIP _____

Phone No. ( ) _____

Method of Payment:

Check ☐ (Please enclose in envelope.)

Charge My: VISA ☐ MasterCard ☐

American Express ☐

Charge # _____

Expiration Date _____

Order No.	Title	Qty.	Price	Total

You can **FAX** your order to **1-317-573-2583**. Or call **1-800-428-5331, ext. ORDR** to order direct.
Please add $2.50 per title for shipping and handling.

Subtotal _____

Shipping & Handling _____

Total _____

QUE

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL
First Class Permit No. 9918 Indianapolis, IN

Postage will be paid by addressee

que®

11711 N. College
Carmel, IN 46032

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL
First Class Permit No. 9918 Indianapolis, IN

Postage will be paid by addressee

que®

11711 N. College
Carmel, IN 46032

Special Edition

MS-DOS® USER'S GUIDE

Maximize your personal computer proficiency with *MS-DOS User's Guide*, Special Edition! Complete with coverage of all DOS commands, utilities, and configuration capabilities, this comprehensive text is the *ultimate* DOS resource!

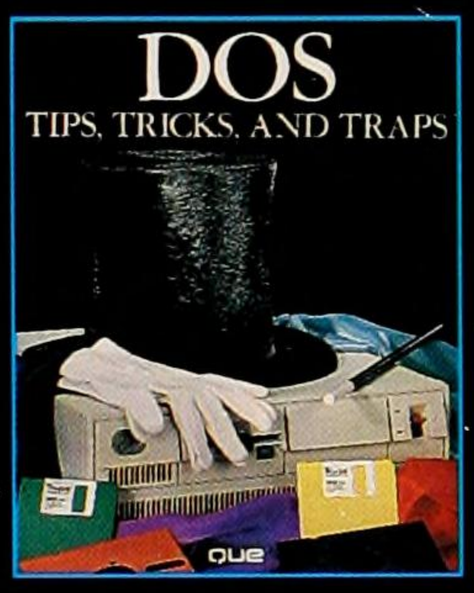
MS-DOS User's Guide, Special Edition, provides a thorough analysis of the DOS Shell, the EDLIN text editor, batch files, tree structure directories, and the ANSI.SYS abilities of DOS. This book includes a complete DOS command summary and numerous appendices provide you with instructions for fixed disk preparation, editing and control key tables, and code page tables.

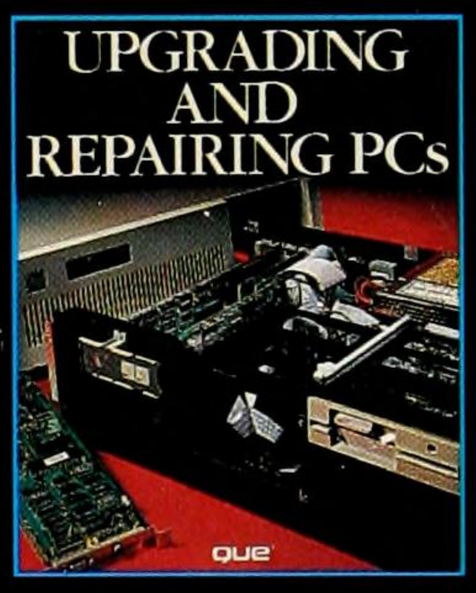
Customize your operating system for powerful results with Que's indespensable *MS-DOS User's Guide*, Special Edition!

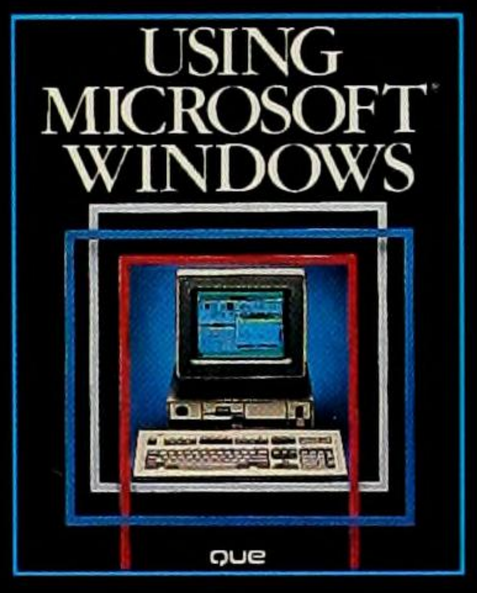
Learn how to . . .

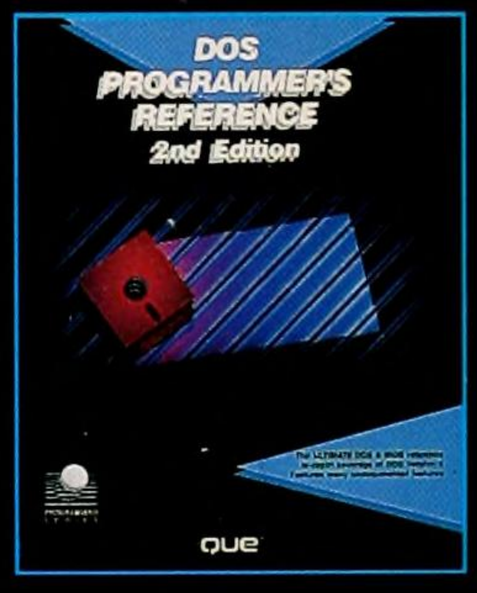
- Create, copy, rename, and erase files
- Streamline system management with tree-structure directories
- Develop batch files and use batch subcommands
- Maximize your hard disk efficiency
- Operate DOS commands in a networked environment

Includes margin notes highlighting tips and techniques









ISBN 0-88022-505-X

1048

$29.95 USA
$39.95 CAN
£27.95 U.K.

IBM/OPERATING SYSTEMS

que®